Πανεπιστήμιο Θεσσαλίας

Τμήμα Μηχ/κών Η/Υ, Τηλεπικοινωνιών & Δικτύων

Διαχείριση και Διάδοση Εμπιστοσύνης σε Ομότιμα Δίκτυα

ΓΚΑΤΖΙΚΗΣ ΛΑΖΑΡΟΣ

ΠΡΟΠΤΥΧΙΑΚΟΣ ΦΟΙΤΗΤΗΣ ΤΜΗΜΑΤΟΣ
ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ : ΚΟΥΤΣΟΠΟΥΛΟΣ ΙΟΡΔΑΝΗΣ

ΒΟΛΟΣ 2006

Ευχαριστίες

Στην προσπάθειά μου αυτή συνέβαλαν πολλοί άνθρωποι τους οποίους νιώθω την ανάγκη να ευχαριστήσω.

Πρώτα απ' όλα θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα της διπλωματικής μου διατριβής, καθηγητή κ . Ιορδάνη Κουτσόπουλο, χωρίς την βοήθεια του οποίου θα ήταν αδύνατη η ολοκλήρωση της εργασίας αυτής.

Επίσης, όλους όσους πορευτήκαμε μαζί αυτά τα χρόνια και με στήριξαν στα προβλήματα που προέκυψαν και με βοήθησαν να εκπληρώσω τους στόχους μου. Εξέχουσα θέση ανάμεσα σε αυτούς κατέχει ο συμφοιτητής και φίλος Τυχογιώργος Γιώργος, τον οποίο θα ήθελα να ευχαριστήσω ονομαστικά.

Τέλος, δεν θα μπορούσα να μην αναφερθώ στην οικογένεια μου, που είναι πάντοτε στο πλευρό μου και στηρίζουν αγόγγυστα τις επιλογές μου.

Σύνοψη

Τα τελευταία χρόνια παρατηρείται ταχύτατη διάδοση των **Peer to Peer (P2P)** δικτύων και η χρήση τους επεκτείνεται συνεχώς σε νέους τομείς, με πιο γνωστή την χρήση τους για διακίνηση αρχείων. Τα δίκτυα αυτά, σε αντίθεση με την καθιερωμένη αρχιτεκτονική πελάτη - εξυπηρετητή (client-server), όπου υπάρχει ένας κεντρικός εξυπηρετητής με τον οποίο επικοινωνούν όλοι οι πελάτες, αποτελούνται από υπολογιστές, που λέγονται peers και δρουν ταυτόχρονα ως πελάτες και ως εξυπηρετητές. Κάθε χρήστης, ως πελάτης παράγει αιτήσεις, με κάποιο ρυθμό, τις οποίες διαχωρίζει ("σπάει") στους υπόλοιπους εξυπηρετητές. Ως εξυπηρετητής κάθε χρήστης εξυπηρετεί τις εισερχόμενες αιτήσεις βάσει μιας πολιτικής προτεραιοτήτων.

Ένα βασικό κριτήριο απόδοσης των δικτύων αυτών είναι η μέση καθυστέρηση απόκτησης ενός αρχείου, η οποία εξαρτάται από τον διαχωρισμό των αιτήσεων στους κόμβους άλλα και από την πολιτική εξυπηρέτησης κάθε εξυπηρετητή. Σκοπός της εργασίας αυτής είναι η λύση του προβλήματος ελαχιστοποίησης της μέσης βεβαρημένης καθυστέρησης στο δίκτυο, όπου η καθυστέρηση κάθε χρήστη εκφράζεται είτε ως η μέση καθυστέρηση του χρήστη στο σύστημα είτε ως η το μέγιστο εκ των καθυστερήσεων σε κάποιο εξυπηρετητή. Η πρώτη περίπτωση είναι μια συνηθισμένη μετρική απόδοσης στην οποία θα χρησιμοποιήσουμε τον cμ rule για τον καθορισμό των προτεραιοτήτων από τους εξυπηρετητές. Η δεύτερη εκφράζει την περίπτωση παράλληλης εξυπηρέτησης από τους διάφορους εξυπηρετητές και μελετούμε αν ισχύει ο cμ rule. Για την εύρεση του βέλτιστου τρόπου διαχωρισμού των αιτήσεων παρουσιάζουμε έναν κατανεμημένο εγωιστικό αλγόριθμο, όπου κάθε χρήστης ελαχιστοποιεί την δική του καθυστέρηση.

Ο αλγόριθμος αυτός βασίζεται στην τεχνική του waterfilling και παρουσιάζουμε τον τρόπο με τον οποίο είναι δυνατή η υλοποίησή του με κατανεμημένο τρόπο και με ελάχιστη ανταλλαγή μηνυμάτων ανάμεσα στους κόμβους.

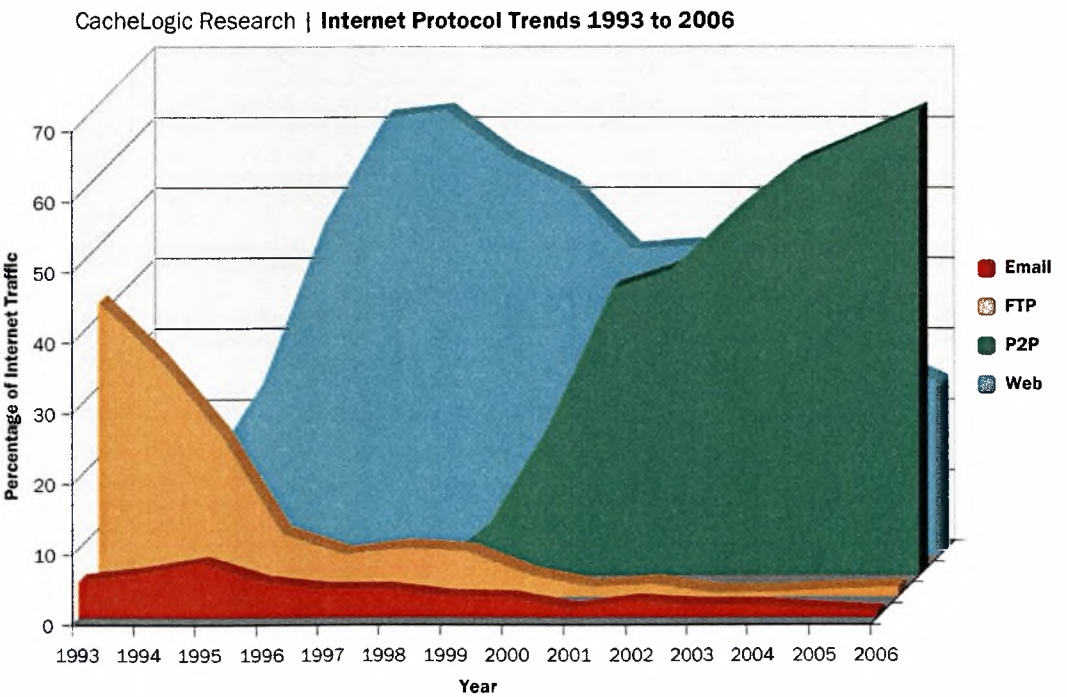
Η εργασία οργανώνεται ως εξής. Στο κεφάλαιο 1 κάνουμε μια εισαγωγή στα P2P δίκτυα παρουσιάζοντας τα κύρια χαρακτηριστικά και εφαρμογές τους. Στο κεφάλαιο 2 παρουσιάζουμε προηγούμενες εργασίες για τα δίκτυα αυτά. Στο κεφάλαιο 3 περιγράφουμε το μοντέλο του συστήματος. Τα κεφάλαια 4 και 5 περιλαμβάνουν την περιγραφή και ανάλυση των προτεινόμενων αλγορίθμων για τον υπολογισμό της καθυστέρησης ως ο μέσος όρος ή το μέγιστο της καθυστέρησης σε κάθε εξυπηρετητή αντίστοιχα . Τέλος, στο κεφάλαιο 6 αναφέρονται τα συμπεράσματα από την παρούσα εργασία.

Κεφάλαιο 1 - Εισαγωγή

Ένα P2P (peer-to-peer) δίκτυο είναι ένα δίκτυο διασυνδεδεμένων υπολογιστών που δεν ακολουθεί την συνηθισμένη αρχιτεκτονική πελάτη-εξυπηρετητή, όπου έχουμε κάποιους εξειδικευμένους υπολογιστές που εξυπηρετούν τις αιτήσεις των υπολοίπων. Πρόκειται για ομότιμα δίκτυα στα οποία κάθε χρήστης λειτουργεί ταυτόχρονα ως πελάτης και ως εξυπηρετητής.

Το μοντέλο αυτό, λοιπόν, είναι διαφορετικό από αυτό που έχουμε συνηθίσει τα τελευταία χρόνια και το οποίο βασίζεται στην ύπαρξη ενός σταθερού εξυπηρετητή και αποτελεί τον κορμό του Internet όπως το γνωρίζουμε σήμερα. Σε γενικές γραμμές, με χαρακτηριστικό παράδειγμα το http, οι χρήστες του Internet ζητούν κάποιο αντικείμενο από κάποιον εξυπηρετητή, ο οποίος λειτουργεί ειδικά για το σκοπό αυτό. Σε ένα P2P δίκτυο, όλοι οι χρήστες συμμετέχουν ενεργά στο σύστημα παρέχοντας πόρους, όπως bandwidth, αποθηκευτικό χώρο και επεξεργαστική ισχύ. Το γεγονός αυτό έχει ως αποτέλεσμα η αύξηση του μεγέθους του δικτύου να αυξάνει και την διαθεσιμότητα πόρων.

Τα τελευταία χρόνια η διάδοση των P2P δικτύων είναι ταχύτατη, με πιο σημαντική και μαζική την χρήση τους για τη διακίνηση αρχείων. Ενδεικτικό της αποδοχής των εφαρμογών αυτών είναι το γεγονός ότι, όπως φαίνεται στο παρακάτω γράφημα, στις μέρες μας η P2P κίνηση καταλαμβάνει περίπου το 60% της συνολικής κίνησης στο Internet. Επίσης, είναι πασιφανής ο ταχύτατος ρυθμός διάδοσης των εφαρμογών αυτών, ενώ βλέπουμε ότι έχουν υποσκελίσει παραδοσιακές χρήσεις του Internet όπως το world wide web.



Εικόνα 1: Κατανομή διαδικτύακης κίνησης

Παρότι όμως η διακίνηση αρχείων είναι η πιο διαδεδομένη εφαρμογή των P2P δικτύων δεν είναι και η μοναδική. Στη συνέχεια περιγράφουμε αναλυτικότερα κάποιες από τις εφαρμογές τους.

- **Content Distribution**

Όπως αναφέρθηκε και προηγουμένως πρόκειται για την πιο διαδεδομένη εφαρμογή των P2P δικτύων. Τα προγράμματα αυτά δίνουν την δυνατότητα στους χρήστες να ανταλλάσσουν δεδομένα με μεγάλη ταχύτητα και παρέχουν αυξημένη διαθεσιμότητα. Στην κατηγορία αυτή περιλαμβάνονται οι γνωστές εφαρμογές διαμοιρασμού αρχείων καθώς και πιο εξεζητημένες εφαρμογές που επιτρέπουν την κατανεμημένη αποθήκευση σημαντικών αρχείων με αποτέλεσμα την καλύτερη και ασφαλέστερη διαχείριση τους. Χαρακτηριστικός εκπρόσωπος αυτής της κατηγορίας αυτής είναι το BitTorrent.

- **Communication and Collaboration**

Στην κατηγορία αυτή περιέχονται τα προγράμματα chat και instant messaging, που επιτρέπουν, συνήθως σε πραγματικό χρόνο, την συνεργασία και την επικοινωνία ανάμεσα στους χρήστες. Χαρακτηριστικό παράδειγμα είναι το Skype.

- **Distributed Computing**

Στην κατηγορία αυτή περιέχονται προγράμματα που στοχεύουν στην αξιοποίηση της υπολογιστικής ισχύος των peers. Χρησιμοποιούνται για την επίλυση πολύπλοκων προβλημάτων τα οποία είναι διαχωρίσιμα σε απλούστερα, τα οποία επιλύονται από κάθε peer, ο οποίος στη συνέχεια επιστρέφει τα αποτελέσματα. Σε αυτές τις εφαρμογές απαιτείται η ύπαρξη κεντρικού ελέγχου για τον διαμοιρασμό των εργασιών και τη συλλογή των αποτελεσμάτων. Χαρακτηριστικό παράδειγμα είναι το Folding@home που ασχολείται με την προσομοίωση της αναδίπλωσης των πρωτεϊνών.

- **P2P Streaming**

Στην κατηγορία αυτή περιέχονται προγράμματα που χρησιμοποιούν P2P πρωτόκολλα για το streaming δεδομένων ήχου και εικόνας. Πρόκειται στην ουσία για την δημιουργία τηλεοπτικών και ραδιοφωνικών σταθμών που εκμεταλλεύονται την P2P αρχιτεκτονική για την εκπομπή του προγράμματος τους. Ενδεικτικό παράδειγμα τέτοιου προγράμματος είναι το PPLive.

Κεφάλαιο 2 – Βιβλιογραφική Έρευνα

Το γεγονός ότι τα Peer-to-Peer δίκτυα χρησιμοποιούνται σήμερα σε πολλές διαφορετικές εφαρμογές καθιστά δύσκολη τη μοντελοποίησή τους. Επιπλέον, η έμφυτη πολυπλοκότητα που παρουσιάζουν λόγω του μεγέθους τους αποτελεί μια ακόμη πρόκληση. Μέχρι σήμερα έχουν γίνει αρκετές αξιόλογες προσπάθειες μοντελοποίησης των Peer-to-Peer δικτύων που αξίζει να αναφέρουμε.

Στο [1] οι συγγραφείς εξετάζουν το πρόβλημα της βέλτιστης επιλογής κόμβων σε δύο πολύ σημαντικές περιπτώσεις: το κατέβασμα ενός αρχείου και τη αναπαραγωγή ενός αρχείου βίντεο ή ήχου που βρίσκεται αποθηκευμένο σε κάποιο άλλο υπολογιστή (streaming). Οι συγγραφείς οραματίζονται ένα Peer-to-Peer δίκτυο που θα λειτουργεί σαν μια αγορά πόρων όπου κάθε χρήστης πουλά τους πόρους που διαθέτει και αγοράζει πόρους από άλλους χρήστες με βάση κάποιες συναρτήσεις κόστους. Υπό αυτές τις συνθήκες, το πρόβλημα της βέλτιστης επιλογής κόμβων αναφέρεται στην επιλογή εκείνων των κόμβων που διαθέτουν το ζητούμενο αρχείο αλλά και εκείνου του ρυθμού κατεβάσματος από κάθε κόμβο ώστε να ελαχιστοποιείται το κόστος απόκτησής του. Ειδικά, στην περίπτωση του streaming το πρόβλημα λύνεται χρησιμοποιώντας τον περιορισμό ότι θα πρέπει να εξασφαλίζεται συνεχής αναπαραγωγή του αρχείου ακόμη και αν αποτύχουν οι συνδέσεις με κάποιο αριθμό κόμβων.

Στο [2] εξετάζεται με βάση απλά μοντέλα το δημοφιλές πρωτόκολλο ανταλλαγής αρχείων BitTorrent ως προς την αποδοτικότητά του. Ελέγχεται η αποτελεσματικότητα του ενσωματωμένου μηχανισμού προσφοράς κινήτρων στους χρήστες ώστε να συνεισφέρουν στο δίκτυο, η αποτελεσματικότητα με την οποία διανέμονται τα αρχεία στους χρήστες αλλά και ο τρόπος που συμπεριφέρεται το πρωτόκολλο καθώς αυξάνεται πολύ ο αριθμός των κόμβων στο δίκτυο. Όσον αφορά την αποτελεσματικότητα στον διαμοιρασμό των αρχείων, οι συγγραφείς προσδιορίζουν την πιθανότητα ένας κόμβος να βρει τα κομμάτια ενός αρχείου που χρειάζεται με βάση κάποιο μέγιστο αριθμό συνδέσεων και αποδεικνύουν ότι αυτή είναι ανεξάρτητη από το ρυθμό άφιξης νέων αιτήσεων και, στις περισσότερες περιπτώσεις, πολύ κοντά στη μονάδα.

Οι συγγραφείς του [3] προτείνουν έναν απλό πρωτόκολλο διανομής αρχείων όπου κάθε αρχείο χωρίζεται σε κομμάτια και κάθε χρήστης που κατεβάζει κάθε κομμάτι μπορεί να το δίνει σε άλλους χωρίς να είναι απαραίτητο να έχει πάρει όλο το αρχείο. Το πρωτόκολλο αποτελείται από ένα συνδυασμό push και pull μηχανισμών, που εφαρμόζονται από κάθε κόμβο σε διαφορετικό στάδιο της διανομής του αρχείου ώστε να επιτευχθεί καλύτερη απόδοση. Επίσης, τονίζουν τη μεγάλη σημασία της σωστής επιλογής του κομματιού που θα ζητήσει ένας κόμβος κάθε στιγμή στην ελαχιστοποίηση του χρόνου διανομής του αρχείου. Τέλος, προσδιορίζεται ο χρόνος που απαιτείται από το πρωτόκολλο για τη διανομή ενός μεγάλου μέρους του αρχείου σε όλους τους χρήστες και αποδεικνύεται πόσο βοηθά τη διαδικασία διανομής το σπάσιμο του αρχείου σε κομμάτια.

Στο [4] εξετάζεται το πρόβλημα της δρομολόγησης διαφορετικών κατηγοριών από πελάτες σε πολλούς κατανεμημένους εξυπηρετητές ώστε να ελαχιστοποιηθεί ο μέσος χρόνος ανάκτησης κάθε κατηγορίας πελατών. Το πρόβλημα αυτό βρίσκει εφαρμογή σε πολλά σύγχρονα κατανεμημένα συστήματα. Στο πρόβλημα της εύρεσης των βέλτιστων προτεραιοτήτων στο δίκτυο αποδεικνύεται ότι η βέλτιστη επιλογή είναι αυτή που επιβάλλεται από τον μC κανόνα. Επίσης, το παραπάνω πρόβλημα ορίζεται ως ένα μη-γραμμικό πρόβλημα βελτιστοποίησης και αποδεικνύοντας ότι κάθε εσωτερικό τοπικό ελάχιστο αποτελεί και ολικό ελάχιστο διευκολύνεται η διαδικασία

επίλυσής του. Τέλος, με βάση το μαθηματικό μοντέλο προτείνονται κάποιοι απλοί προσεγγιστικοί αλγόριθμοι με απόδοση πολύ κοντά στα θεωρητικά αποτελέσματα.

Στο [5] ο συγγραφέας αναλύει βασικές έννοιες των ουρών δίνοντας ιδιαίτερη έμφαση σε θέματα απόδοσης των συστημάτων που καθορίζονται από παράγοντες όπως οι φυσικές παράμετροι του συστήματος (ταχύτητα εξυπηρέτησης, αριθμός εξυπηρετητών κ.α.), τα χαρακτηριστικά της κίνησης (η κατανομή των αφίξεων, τα μεγέθη των αιτημάτων κ.α.) και ο τρόπος εξυπηρέτησης των αιτήσεων. Ειδικά για την περίπτωση της M/G/1 ουράς, αποδεικνύονται οι τύποι που δίνουν τους χρόνους αναμονής και ανάκτησης ενώ εξετάζεται και η περίπτωση των ουρών πολλαπλών κλάσεων με προτεραιότητες. Ακόμη, καλύπτονται οι περιπτώσεις των preemptive αλλά και non-preemptive μεθόδων παραχώρησης προτεραιότητας καθώς και η περίπτωση της γενικής ουράς G/G/1.

Στο [6] γίνεται μια προσπάθεια κατανόησης της Peer-to-Peer εφαρμογής ανταλλαγής αρχείων KaZaA. Το KaZaA είναι σήμερα μια από τις σημαντικότερες εφαρμογές του internet με περισσότερους από 3 εκατομμύρια χρήστες. Όμως το γεγονός ότι πρόκειται για ένα εμπορικό πρόγραμμα που χρησιμοποιεί κρυπτογράφηση είναι πολύ δύσκολο να μάθουμε τον τρόπο λειτουργίας του. Οι συγγραφείς σε μια προσπάθεια να αποκαλυφθούν τα βασικά χαρακτηριστικά του KaZaA χρησιμοποίησαν κάποια εργαλεία παρακολούθησης δικτύου ώστε να μελετήσουν τα μηνύματα που ανταλλάσσονται μεταξύ των κόμβων από το KaZaA. Με τον τρόπο αυτό κατάφεραν να βγάλουν σημαντικά συμπεράσματα για τη δομή και την λειτουργία του.

Οι συγγραφείς του [7] πραγματεύονται το πρόβλημα της διανομής ενός αρχείου σε ένα δίκτυο όπου υπάρχουν κόμβοι που είτε το έχουν ολόκληρο (seeders) είτε έχουν ένα μέρος του και θέλουν να κατεβάσουν και το υπόλοιπο (leechers), οι οποίοι όμως συνεισφέρουν και αυτοί στο διαμοιρασμό του. Σκοπός είναι η διανομή ολόκληρου το αρχείου στους leechers στο λιγότερο δυνατό χρόνο. Στη δημοσίευση αυτή παρουσιάζονται τύποι για τον ελάχιστο χρόνο διανομής του αρχείου σε ένα τέτοιο δίκτυο, οι οποίοι επαληθεύονται και πρακτικά με παρατήρηση πραγματικών δικτύων. Οι τύποι αυτοί εξαρτώνται από το μέγεθος του αρχείου, τους ρυθμούς αποστολής δεδομένων των seeders και τους ρυθμούς με τους οποίους μπορούν να ανεβάζουν και να κατεβάζουν δεδομένα οι leechers, Επίσης, εξετάζεται η περίπτωση όπου οι κόμβοι που θέλουν να κατεβάσουν το αρχείο χωρίζονται σε δύο κατηγορίες με διαφορετική προτεραιότητα. Συγκεκριμένα, εξετάζεται ο τρόπος με τον οποίο οι κόμβοι με την μικρότερη προτεραιότητα μπορούν να συνεισφέρουν σημαντικά στην ταχύτερη διανομή του αρχείου στους κόμβους με μεγαλύτερη προτεραιότητα.

Ο συγγραφέας του [15] εξετάζει το πρόβλημα ελαχιστοποίησης της μέσης και της μέγιστης καθυστέρησης ανάκτησης ενός αρχείου χρησιμοποιώντας κατανεμημένες υλοποιήσεις που βασίζονται στην αλτρουιστική συμπεριφορά των κόμβων. Στην παρούσα εργασία συγκρίνουμε την προσέγγιση αυτή με την εγωιστική προσέγγιση που προτείνουμε.

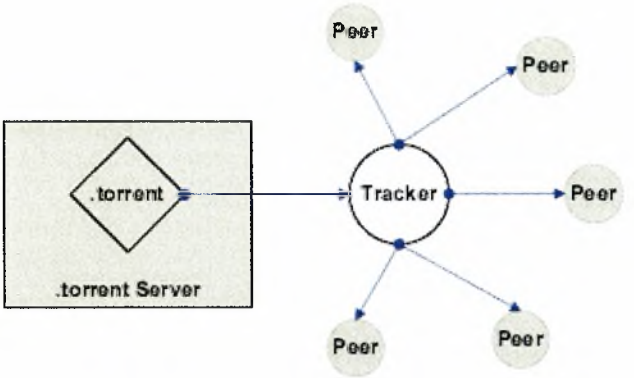
Στην συνέχεια θα παρουσιάσουμε αναλυτικά το πρωτόκολλο BitTorrent καθώς οι προτεινόμενοι αλγόριθμοι μπορούν να υλοποιηθούν και να ενσωματωθούν εύκολα στο πρωτόκολλο αυτό.

Το BitTorrent αποτελεί ένα Peer – to – Peer πρωτόκολλο δεύτερης γενιάς που αναπτύχθηκε το 2002 από τον Bram Cohen για να παρέχει γρήγορο διαμοιρασμό κυρίως δημοφιλών αρχείων. Τυπικά, για ένα δημοφιλές αρχείο υπάρχουν μερικές χιλιάδες κόμβων που το κατεβάζουν ταυτόχρονα ενώ κατά τη διάρκεια ζωής του ο συνολικός αριθμός χρηστών που θα το κατεβάσουν μπορεί να είναι δεκάδες ή ακόμη και εκατοντάδες χιλιάδες.

Η βασική ιδέα στην οποία στηρίζεται το BitTorrent είναι ο χωρισμός ενός αρχείου σε κομμάτια μεγέθους 256KB το καθένα. Όταν κάποιος κόμβος θέλει, λοιπόν, να κατεβάσει κάποιο αρχείο πρέπει να συνδεθεί ταυτόχρονα με πολλούς κόμβους που έχουν τα κομμάτια του και να τα κατεβάσει. Σύμφωνα με το πρωτόκολλο, υπάρχουν δύο κατηγορίες κόμβων, αυτοί που έχουν όλα τα κομμάτια που αποτελούν το αρχείο, οι οποίοι λέγονται seeders, και εκείνοι που έχουν ορισμένα από τα κομμάτια που το αποτελούν, και λέγονται downloaders. Σκοπός των seeders είναι να παραμένουν στο δίκτυο ώστε να δίνουν τα κομμάτια που αποτελούν το αρχείο σε άλλους, επομένως οι κόμβοι αυτοί μόνο ανεβάζουν δεδομένα. Οι downloaders, αντίθετα, κατεβάζουν αλλά και ανεβάζουν δεδομένα. Όπως ορίζει το πρωτόκολλο, κάθε κόμβος μπορεί να συνεισφέρει στον διαμοιρασμό ενός κομματιού ενός αρχείου μόλις ολοκληρωθεί το κατέβασμα του κομματιού αυτού. Αυτό σημαίνει ότι ένας downloader μπορεί να ανεβάζει όλα τα τμήματα τα οποία έχει κατεβάσει βοηθώντας έτσι το έργο των seeders. Το πρωτόκολλο προσπαθεί, εφαρμόζοντας την πολιτική «Το σπανιότερο κομμάτι πρώτο» να εξασφαλίσει ομοιόμορφή κατανομή των κομματιών στο δίκτυο έτσι ώστε να εξασφαλίζεται ομαλός διαμοιρασμός όλων των κομματιών ενός αρχείου.

Για την διευκόλυνση αυτής της διαδικασίας, το BitTorrent χρησιμοποιεί ένα μηχανισμό που ονομάζεται tracker και που φαίνεται σχηματικά στην Εικόνα 2. Όταν ένας κόμβος θέλει να κατεβάσει ένα αρχείο πρέπει με κάποιο τρόπο να συνδεθεί στον tracker του αρχείου αυτού. Για να το κάνει αυτό, κατεβάζει ένα αρχείο με κατάληξη .torrent. Το αρχείο αυτό περιέχει πληροφορίες σχετικά με το αρχείο, πρόκειται δηλαδή για ένα meta-data αρχείο. Οι πληροφορίες που περιέχει είναι το μέγεθος του αρχείου, το όνομά του, κάποιες πληροφορίες κατακερματισμού, και τη διεύθυνση του tracker. Ο tracker διατηρεί καταγεγραμμένους όλους τους κόμβους που έχουν το αρχείο αυτό, είτε ολόκληρο είτε ένα τμήμα του και χρησιμοποιεί ένα απλό πρωτόκολλο που βασίζεται στο HTTP. Με βάση αυτό κάθε κόμβος στέλνει ένα μήνυμα με πληροφορίες για το αρχείο που κατεβάζει και έναν αριθμό port στην οποία θα γίνουν οι συνδέσεις. Στη συνέχεια, ο tracker απαντά με μια τυχαία λίστα των κόμβων που έχουν κατεβάσει ή κατεβάζουν εκείνη τη στιγμή το αρχείο. Ο κόμβος τότε συνδέεται με αυτούς και μαθαίνει ποια κομμάτια του αρχείου έχει κάθε ένας. Έπειτα, ο κόμβος ζητά από τους άλλους όλα τα κομμάτια τα οποία δεν έχει. Όταν τελειώσει το κατέβασμα ενός κομματιού εφαρμόζει σε αυτό μια συνάρτηση κατακερματισμού για να ελέγξει αν το κατέβασε σωστά και στη συνέχεια ανακοινώνει σε όλους τους κόμβους με τους οποίους είναι συνδεδεμένος ότι έχει πάρει το κομμάτι αυτό και μπορεί στο εξής να το ανεβάζει και σε άλλους. Κάθε κόμβος μπορεί να στέλνει δεδομένα ταυτόχρονα σε περιορισμένο αριθμό χρηστών, που συνήθως ορίζεται στους τέσσερις. Οι κόμβοι που θα επιλεγούν για ανέβασμα καθορίζονται από τον ρυθμό κατεβάσματος από αυτούς εκείνη τη στιγμή. Κάθε κόμβος ανεβάζει στους τέσσερις κόμβους οι οποίοι του δίνουν δεδομένα με μεγαλύτερο ρυθμό. Οι αιτήσεις για ανέβασμα που φτάνουν από άλλους κόμβους μπαίνουν σε μία ουρά και εξυπηρετούνται αργότερα. Με τον τρόπο αυτό ενθαρρύνονται οι κόμβοι να ανεβάζουν δεδομένα, αντιμετωπίζοντας έτσι το φαινόμενο free-riding, όπου ένας

κόμβος ενδιαφέρεται μόνο να κατεβάσει ένα αρχείο χωρίς να βοηθά και ο ίδιος στο διαμοιρασμό του.



Εικόνα 2: Η αρχιτεκτονική BitTorrent αποτελούμενη από τον tracker, το αρχείο .torrent και τους κόμβους

Το φαινόμενο κατά το οποίο ένας κόμβος αρνείται προσωρινά να ανεβάσει δεδομένα σε κάποιους άλλους λέγεται choking. Αυτό μπορεί να συμβεί για πολλούς λόγους, όπως το γεγονός ότι το πρωτόκολλο TCP δεν αποδίδει καλά όταν υπάρχουν πολλές συνδέσεις, οπότε πρέπει να υπάρχει ένας μέγιστος αριθμός συνδέσεων. Είναι πολύ σημαντικό για κάποιο κόμβο να εφαρμόζει ένα καλό αλγόριθμο choking έτσι ώστε να αποφεύγονται συνεχή ανοίγματα και κλεισίματα συνδέσεων, καθώς καταναλώνουν μέρος των πόρων, αλλά και να εξασφαλίζεται σύνδεση σε κόμβους που θα του επιτρέψουν να κατεβάσει με το μεγαλύτερο δυνατό ρυθμό. Για την επίτευξη του τελευταίου κάθε κόμβος εφαρμόζει έναν αλγόριθμο, που ονομάζεται optimistic unchoking. Αφού κάθε κόμβος, έστω ο Α, ανεβάζει μόνο σε τέσσερις κόμβους, εκείνους που το δίνουν εκείνη τη στιγμή με τη μεγαλύτερη ταχύτητα, είναι δυνατό να υπάρχει κάποιος άλλος κόμβος Β που αν ανέβαζε σε αυτόν ο Α, να μπορούσε να του εξασφαλίσει μεγαλύτερο ρυθμό κατεβάσματος από κάθε άλλο κόμβο στον οποίο ανεβάζει αυτή τη στιγμή ο Α. Για να μπορεί, λοιπόν, κάθε κόμβος να ανακαλύπτει το ρυθμό κατεβάσματος από άλλους κόμβους επιλέγει τυχαία έναν πέμπτο κόμβο, που του έχει ζητήσει κάποιο κομμάτι που διαθέτει, και τον εξυπηρετεί. Αφού αντιληφθεί το ρυθμό με τον οποίο μπορεί αυτός ο κόμβος να του δίνει δεδομένα, διακόπτει το ανέβασμα σε εκείνο από τους πέντε με το μικρότερο ρυθμό. Αυτή η διαδικασία επαναλαμβάνεται κάθε 30 δευτερόλεπτα και έτσι επιτυγχάνεται ο μέγιστος ρυθμός κατεβάσματος.

Η αναζήτηση των αρχείων καθώς και η διανομή των .torrent αρχείων γίνεται μέσω δικτυακών τόπων (SuprNova.org, PirateBay.org, TorrentSpy.com κ.α). Κάθε χρήστης μπορεί να αναζητήσει το αρχείο που θέλει, να κατεβάσει το .torrent αρχείο που αναφέρεται σε αυτό και να το ανοίξει με ένα από τα πολλά διαθέσιμα συμβατά με το πρωτόκολλο προγράμματα (μTorrent, BitComet, ShareAza κ.α.). Στη συνέχεια το πρόγραμμα θα αναλάβει να συνδεθεί με τον tracker και να κατεβάσει όλα τα κομμάτια από τα οποία αποτελείται το αρχείο.

Σύμφωνα με μετρήσεις της κίνησης στις κεντρικές αρτηρίες του Internet τον Ιούνιου 2004, το ποσοστό της κίνησης που χρησιμοποιεί το πρωτόκολλο BitTorrent ανερχόταν στο 53% της συνολικής P2P κίνησης. Η τεράστια αυτή διάδοση του BitTorrent οφείλεται σε κάποια πολύ σημαντικά πλεονεκτήματα που έχει. Το βασικό πλεονέκτημά του είναι το γεγονός ότι στο διαμοιρασμό των αρχείων μπορούν να συνεισφέρουν ακόμη και οι κόμβοι που δεν έχουν ολοκληρώσει το κατέβασμά του. Με τον τρόπο αυτό μειώνονται οι απαιτήσεις σε bandwidth από τους seeders, αφού

υποβοηθούνται και από τους leechers, αλλά και γίνεται και γρηγορότερος ο διαμοιρασμός του αρχείου, αφού όταν ο seeder δίνει ένα κομμάτι σε κάποιο downloader αυξάνει ταυτόχρονα και την διαθεσιμότητά του. Επιπλέον, η χρήση συναρτήσεων κατακερματισμού για τον έλεγχο των κομματιών βοηθά στο σωστότερο διαμοιρασμό των αρχείων.

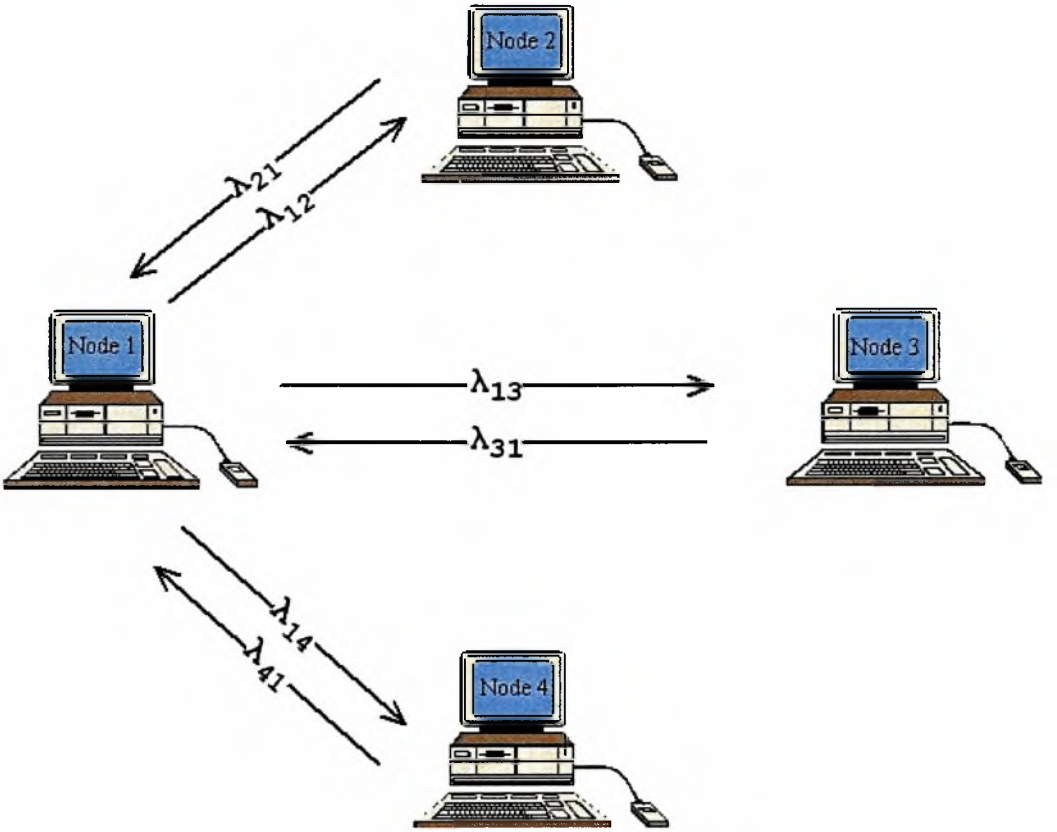
Κεφάλαιο 3 – Το Μοντέλο του συστήματος

Θεωρούμε ένα peer-to-peer σύστημα που αποτελείται από N κόμβους, καθένας εκ των οποίων έχει ένα σύνολο αρχείων προς διάθεση. Τα αρχεία αυτά μπορεί να είναι οποιασδήποτε μορφής και το μέγεθος τους γενικά κυμαίνεται από μερικά KB μέχρι και μερικά GB. Η σύσταση του συστήματος είναι σταθερή, θεωρούμε δηλαδή ότι δεν εισέρχονται ούτε αποχωρούν κόμβοι από το σύστημα. Κάθε κόμβος προσφέρει ένα αρχείο του απαντώντας σε αιτήσεις των άλλων χρηστών ενώ ταυτόχρονα ζητά αρχεία από τους άλλους. Κάθε κόμβος λοιπόν, λειτουργεί ταυτόχρονα και ως εξυπηρετητής και ως πελάτης. Αυτό το σύνολο κόμβων θα μπορούσε να προκύπτει από ένα μηχανισμό εύρεσης αρχείου ενός peer-to-peer προγράμματος, όπως ο μηχανισμός του BitTorrent .

Συμβολίζουμε τον συνολικό ρυθμό παραγωγής αιτήσεων του κόμβου i με λ_i που δηλώνει τον αριθμό των αιτήσεων για διαφορετικά αντικείμενα ανά μονάδα χρόνου. Θεωρούμε ότι η παραγωγή αιτήσεων κάθε κόμβου i ακολουθεί κατανομή Poisson, ενώ το μέγεθος των αιτήσεων είναι εκθετική τυχαία μεταβλητή με μέση τιμή s_i. Επομένως ο συνολικός ρυθμός παραγωγής αιτήσεων κάθε κόμβου i σε bits/sec είναι $\lambda_i s_i$. Οι N κόμβοι σχηματίζουν ένα κλειστό δίκτυο, όπου κάθε αίτηση ενός κόμβου μπορεί να εξυπηρετηθεί από ένα υποσύνολο των άλλων κόμβων. Συμβολίζουμε, λοιπόν, με λ_{ij} το ρυθμό παραγωγής αιτήσεων του i που ανατίθεται στον j. Ο διαμοιρασμός του ρυθμού παραγωγής αιτήσεων του πελάτη i μπορεί να αναπαρασταθεί με ένα διάνυσμα $\boldsymbol{\lambda}_i = \left(\lambda_{ij} : j = 1,...N, j \neq i\right)$ με $\lambda_i = \sum_{j=1, j\neq i}^{N} \lambda_{ij}$. Εναλλακτικά μπορεί να γραφεί ως ένα διάνυσμα πιθανοτήτων $\mathbf{p}_i = \left(p_{ij} : j = 1,...N, j \neq i\right)$ όπου $p_{ij} = \frac{\lambda_{ij}}{\lambda_i}$ είναι το ποσοστό των αιτήσεων του κόμβου i που στέλνονται στον j. Ο διαμοιρασμός αυτός των αιτήσεων μπορεί επίσης να αναπαριστά την περίπτωση που ένας χρήστης ανακτά τμήματα του ίδιου αντικειμένου από διάφορους κόμβους. Ο διαμοιρασμός των ρυθμών παραγωγής των αιτήσεων όλου του συστήματος μπορεί να αναπαρασταθεί με έναν πίνακα $\boldsymbol{\Lambda}$ που η i-οστή γραμμή του είναι το διάνυσμα $\boldsymbol{\lambda}_i$ και τα στοιχεία της διαγωνίου είναι μηδενικά. Εναλλακτικά, μπορεί να αναπαρασταθεί με ένα πίνακα πιθανοτήτων $\boldsymbol{P}$ που η i-οστή γραμμή του είναι το διάνυσμα $\mathbf{p}_i$.

Κάθε κόμβος j ως εξυπηρετητής δέχεται αιτήσεις, τις οποίες πρέπει να εξυπηρετήσει, με ρυθμό λ_{ij} από τους άλλους κόμβους i≠j. Κάθε εξυπηρετητής j χαρακτηρίζεται από τον ρυθμό εξυπηρέτησης C_j. Επομένως, ο μέσος χρόνος εξυπηρέτησης του i από τον j είναι s_i / C_j.

Το μοντέλο αυτό απεικονίζεται στην παρακάτω εικόνα για ένα δίκτυο τεσσάρων κόμβων.



Εικόνα 3: Το δίκτυο για τέσσερις κόμβους

Ευστάθεια: Η αναγκαία και ικανή συνθήκη για ευστάθεια (δηλαδή, για πεπερασμένη καθυστέρηση) είναι $\sum_{i=1,i\neq j}^{N}\lambda_{ij}s_i < C_j$ για κάθε εξυπηρετητή $j=1,...N$. Το σύστημα είναι ευσταθές εάν $\sum_{i=1}^{N}\lambda_i s_i < \sum_{j=1}^{N}C_j$,συνθήκες οι οποίες θεωρούμε ότι ισχύουν.

Αν υποθέσουμε ότι το μέγεθος αρχείων s που αιτούνται οι πελάτες ακολουθεί εκθετική κατανομή με την ίδια μέση τιμή για όλους, τότε ο μέσος ρυθμός εξυπηρέτησης στον εξυπηρετητή j είναι $\mu_j = s_i / C_j$ και οι χρόνοι εξυπηρέτησης ακολουθούν εκθετική κατανομή. Βάσει αυτής της υπόθεσης, κάθε εξυπηρετητής μπορεί να μοντελοποιηθεί ως μια M/M/1 ουρά με ρυθμό εξυπηρέτησης μ_j. Οι εξυπηρετητές εφαρμόζουν μια preemptive πολιτική εξυπηρέτησης, με το σύνολο της χωρητικότητας του εξυπηρετητή, βάσει προτεραιοτήτων. Η εξυπηρέτηση μιας αίτησης διακόπτεται αν υπάρξει άφιξη αίτησης με μεγαλύτερη προτεραιότητα και συνεχίζεται από το σημείο που διακόπηκε μόλις εξυπηρετηθούν όλοι ο χρήστες μεγαλύτερης προτεραιότητας. Η ανάθεση προτεραιοτήτων από έναν εξυπηρετητή j προσδιορίζεται από ένα διάνυσμα στήλη $\boldsymbol{\pi}_j$. Το j-οστό στοιχείο του διανύσματος είναι μηδέν ενώ τα υπόλοιπα στοιχεία $\pi_j(i)$ αποτελούν μία αναδιάταξη του $\{1,....,N\}\setminus\{j\}$ τέτοια ώστε αν $\pi_j(i) < \pi_j(k)$, τότε ο χρήστης i απολαμβάνει

μεγαλύτερης προτεραιότητας του k στον εξυπηρετητή j. Με ***Π*** συμβολίζουμε τον πίνακα προτεραιοτήτων του δικτύου, που έχει στη στήλη j το διάνυσμα $\boldsymbol{\pi}_j$. Ο μέσος χρόνος ανάκτησης των αιτήσεων του i με προτεραιότητα $\pi_j(i)$ στον εξυπηρετητή j δίνεται από τον τύπο

$$D_{ij} = \frac{1}{\mu_j \left(1 - \sum_{k:\pi_j(k)>\pi_j(i)} \rho_{kj}\right)\left(1 - \sum_{k:\pi_j(k)>\pi_j(i)} \rho_{kj}\right)} \qquad (3.1)$$

όπου $\rho_{ij} = \frac{\lambda_{ij}}{\mu_j}$ είναι ο φόρτος του i στον j. Ο παραπάνω χρόνος ανάκτησης περιλαμβάνει τον χρόνο αναμονής στην ουρά και τον χρόνο εξυπηρέτησης.

Είναι εμφανές ότι η αναμονή D_{ij} εξαρτάται από την προτεραιότητα που απολαμβάνει ο i όταν εξυπηρετείται από τον j, καθώς και από τα σπασίματα λ_{ij}. Η μέση καθυστέρηση του χρήστη i στο σύνολο των άλλων κόμβων είναι $D_i = \sum_{j=1, j\neq i}^{N} p_{ij} D_{ij}$, ενώ η μέγιστη καθυστέρηση είναι $D_i = \max_{\lambda_{ij}} p_{ij} D_{ij}$. Επομένως, κάθε χρήστης μπορεί να επηρεάσει το μέσο χρόνο αναμονής του μόνο μερικώς, ελέγχοντας τα σπασίματα του.

Κεφάλαιο 4 – Ελαχιστοποίηση μέσου χρόνου ανάκτησης

Θεωρούμε ότι σε κάθε κόμβο i αντιστοιχίζεται ένα θετικό βάρος b_i. Αρχικά πραγματευόμαστε την εύρεση ενός εφικτού πίνακα σπασιμάτων $\boldsymbol{\Lambda}$ και ενός πίνακα προτεραιοτήτων $\boldsymbol{\Pi}$ που λύνει το παρακάτω πρόβλημα

$$\min_{\Lambda,\Pi} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} D_i = \min_{\Lambda,\Pi} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} \sum_{j=1, j\neq i}^{N} \frac{\lambda_{ij}}{\lambda_i} D_{ij}(\Lambda,\Pi) = \min_{\Lambda,\Pi} \sum_{j=1}^{N} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) \qquad (4.1)$$

Κάθε εξυπηρετητής επηρεάζει τους χρόνους αναμονής των κόμβων ανάλογα με την προτεραιότητα που δίνει στον καθένα. Κάθε πελάτης i επηρεάζει μερικώς την δικιά του καθυστέρηση επιλέγοντας τα σπασίματα των αιτήσεων του στους εξυπηρετητές. Ωστόσο, με αυτό τον τρόπο επηρεάζει και τις καθυστερήσεις των κόμβων χαμηλότερης προτεραιότητας όπως φαίνεται από την έκφραση του D_{ij}. Είναι εμφανές ότι το παραπάνω πρόβλημα είναι η εύρεση της κοινωνικά βέλτιστης λύσης.

Το παραπάνω πρόβλημα ελαχιστοποίησης μπορεί ισοδύναμα να γραφεί ως:

$$\min_{\Lambda,\Pi} \sum_{j=1}^{N} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) = \min_{\Lambda} \min_{\Pi} \sum_{j=1}^{N} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) = \min_{\Lambda} \sum_{j=1}^{N} \min_{\pi(j)} \sum_{i=1, i\neq j}^{N} b_i \frac{\lambda_i}{\lambda} p_{ij} D_{ij}(\Lambda,\Pi) \qquad (4.2)$$

Με άλλα λόγια, για ένα δεδομένο πίνακα σπασιμάτων, το πρόβλημα της εύρεσης του πίνακα προτεραιοτήτων μπορεί να αναλυθεί σε ανεξάρτητα προβλήματα για κάθε εξυπερετητή. Το καθένα εκ των προβλημάτων είναι η εύρεση του διανύσματος προτεραιοτήτων που ελαχιστοποιεί το βεβαρημένο άθροισμα των χρόνων αναμονής στον συγκεκριμένο εξυπηρετητή.

Η λύση αυτού του προβλήματος δίνεται από τον μc κανόνα. Σύμφωνα με αυτόν, η πολιτική που ελαχιστοποιεί την συνολική βεβαρημένη καθυστέρηση είναι να δίνεις προτεραιότητα κατά φθίνουσα σειρά βάσει του γινομένου βάρους και ρυθμού εξυπηρέτησης, Η απόδειξη βασίζεται στην παρακάτω ιδέα: Ξεκινώντας από μια υποθετικά βέλτιστη λύση εναλλάσσουμε τις προτεραιότητες δύο διαδοχικών, ως προς την προτεραιότητα, πελατών και χρησιμοποιώντας τον Conservation law εξάγουμε την παραπάνω συνθήκη για την βέλτιστη λύση (βλ. [5]). Στην περίπτωσή μας τα βάρη είναι τα b_i και οι ρυθμοί εξυπηρέτησης μ_j.

Βάσει του παραπάνω αποτελέσματος η βέλτιστη ανάθεση προτεραιοτήτων είναι γνωστή, και μάλιστα είναι τέτοια ώστε κάθε χρήστης να απολαμβάνει την ίδια προτεραιότητα σε κάθε εξυπηρετητή, πρόκειται δηλαδή για μια καθολική διάταξη για όλο το σύστημα. Επομένως, το πρόβλημα ελαχιστοποίησης περιορίζεται στην εύρεση του βέλτιστου εφικτού πίνακα σπασιμάτων $\boldsymbol{\Lambda}$.

Το πρόβλημα αυτό μπορεί εύκολα να λυθεί με κεντρικοποιημένο τρόπο. τη συνέχεια θα παρουσιάσουμε δύο αλγορίθμους για την επίλυση του παραπάνω προβλήματος ελαχιστοποίησης με κατανεμημένο τρόπο. Κάθε πελάτης i καθορίζει το δικό του διάνυσμα σπασιμάτων $\boldsymbol{\lambda}_i$ βάσει ενός κριτηρίου βελτιστοποίησης. Οι αλγόριθμοι βασίζονται στην εγωιστική προσέγγιση καθώς κάθε πελάτης επιλέγει τα σπασίματα του ώστε να ελαχιστοποιήσει την δική του καθυστέρηση στο σύστημα αδιαφορώντας για την επίδραση που έχει αυτή η επιλογή στις καθυστερήσεις των χρηστών μικρότερης προτεραιότητας. Για την επίλυση και των δύο προβλημάτων ελαχιστοποίησης, μπορεί να χρησιμοποιηθεί η μέθοδος Waterfilling, την οποία αναλύουμε στο επόμενο εδάφιο.

Η μέθοδος Waterfilling

Η μέθοδος αυτή χρησιμοποιείται για την επίλυση προβλημάτων της μορφής:

$$\min_{\lambda_{ij}} F\left(\lambda_{ij}\right) \qquad (4.3)$$
$$s.t. \qquad \sum_{j} \lambda_{ij} = \lambda_i$$

Δηλαδή πρόκειται για προβλήματα στα οποία θέλουμε να μοιράσουμε κάποια ποσότητα σε N κόμβους, στη συγκεκριμένη περίπτωση το ρυθμό παραγωγής αιτήσεων, ώστε να ελαχιστοποιείται η αντικειμενική συνάρτηση. Η μέθοδος αυτή χρησιμοποιεί τις μερικές παραγώγους $\frac{\partial F_i}{\partial \lambda_{ij}}$ ώστε να λύσει το πρόβλημα.

Αφού ο χρήστης i θέλει να ελαχιστοποιήσει την αντικειμενική του συνάρτηση, θα προσπαθήσει να πάρει μια μικρή ποσότητα ρυθμού ε από τον εξυπηρετητή m και να το κατανείμει στον n έτσι ώστε η αντικειμενική του συνάρτηση να μειωθεί. Μάλιστα, με αυτή την ανακατανομή του ρυθμού ο i θα ήθελε να καταφέρει τη μεγαλύτερη δυνατή μείωση. Για να το καταφέρει αυτό, ο κόμβος i αφαιρεί ποσότητα ίση με ε από τον εξυπηρετητή m, με $m = \arg\max_{j \neq i} \frac{\partial F_i}{\partial \lambda_{ij}}$, και το κατανέμει στον n, με $n = \arg\min_{j \neq i} \frac{\partial F_i}{\partial \lambda_{ij}}$. Με άλλα λόγια, αφαιρείται ποσότητα ε από τον εξυπηρετητή που αντιστοιχεί στο μικρότερο ρυθμό μείωσης και προστίθεται σε εκείνον που θα προσφέρει τη μεγαλύτερη δυνατή μείωση στην αντικειμενική συνάρτηση. Η διαδικασία αύτη εξασφαλίζει ότι μειώνεται πάντα η αντικειμενική συνάρτηση.

Ο κόμβος i συνεχίζει τη διαδικασία ανακατανομής του ρυθμού παραγωγής αιτήσεων μέχρι να μην είναι δυνατή καμία περαιτέρω μείωση της αντικειμενικής συνάρτησης. Στο σημείο εκείνο, όλες οι μερικές παράγωγοι της αντικειμενικής συνάρτησης θα είναι ίσες μεταξύ τους και τότε θα έχει βρεθεί το βέλτιστο σημείο.

Τέλος, είναι σημαντικό να τονίσουμε τη σπουδαιότητα της ακρίβειας ε που χρησιμοποιούμε στον αλγόριθμο. Όσο πιο μεγάλη τιμή βάλουμε στο ε τόσο πιο γρήγορα θα συγκλίνει ο αλγόριθμος, αφού θα πλησιάζει γρηγορότερα στη βέλτιστη λύση, αλλά θα χάσουμε σε ακρίβεια. Αντίθετα, όταν επιλέξουμε μικρή τιμή για το ε ο αλγόριθμος θα χρειαστεί περισσότερες επαναλήψεις αλλά το αποτέλεσμα θα είναι πιο κοντά στην πραγματική βέλτιστη λύση.

Στη συνέχεια παρουσιάζουμε τις προσεγγίσεις αυτές και δείχνουμε πως μπορούν να εφαρμοστούν σε κατανεμημένα πρωτόκολλα με την κατάλληλη ανταλλαγή μηνυμάτων ανάμεσα στους χρήστες. Για απλοποίηση των υπολογισμών από εδώ και στο εξής θεωρούμε ότι ο μέσος ρυθμός εξυπηρέτησης σε όλους τους εξυπηρετητές είναι $\mu = 1$. Επίσης, για ευκολία και χωρίς απώλεια της γενικότητας μπορούμε να θεωρήσουμε ότι οι κόμβοι ονοματίζονται βάσει της προτεραιότητας που απολαμβάνουν στο σύστημα, με τον πρώτο κόμβο να είναι αυτός με την μεγαλύτερη προτεραιότητα. Τέλος, θεωρούμε ότι κάθε αίτηση ενός κόμβου μπορεί να εξυπηρετηθεί από οποιοδήποτε άλλο κόμβο. Αυτό ισχύει στην περίπτωση που το δίκτυο αποτελείται μόνο από κόμβους που διαθέτουν ένα σημαντικό ποσοστό του αρχείου, τέτοιο ώστε κάθε άλλος κόμβος να έχει τη δυνατότητα να κατεβάσει τμήματα του από οποιονδήποτε άλλο.

Εγωιστική Προσέγγιση

Περιγραφή Μεθόδου

Κάθε χρήστης i επιλέγει τα σπασίματα του ώστε να ελαχιστοποιήσει την δική του καθυστέρηση στο σύστημα, προσπαθεί δηλαδή να επιλύσει το παρακάτω πρόβλημα:

$$\min_{\lambda_{ij}} b_i D_i = \min_{\lambda_{ij}} \frac{b_i}{\lambda_i} \sum_{j=1, j\neq i}^{N} F_{ij}\left(\lambda_{ij}\right) \tag{4.4}$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{N} \lambda_{ij} = \lambda_i \qquad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} > 0 \qquad \forall i, j$$

με $F_{ij}(\lambda_{ij}) = \lambda_{ij} \frac{1}{(1-\sum_{k\leq i}\lambda_{kj})(1-\sum_{k<i}\lambda_{kj})}$

Εστιάζοντας λοιπόν στον χρήστη i, είναι εμφανές ότι η $F_{ij}\left(\lambda_{ij}\right)$,είναι κυρτή συνάρτηση του λ_{ij} και καθώς το $\frac{b_i}{\lambda_i}$ είναι ανεξάρτητο των λ_{ij}, σκοπός του χρήστη i είναι η ελαχιστοποίηση υπό τους παραπάνω περιορισμούς, της $F_i = \sum_{j=1, j\neq i}^{N} F_{ij}$, που είναι κυρτή ως άθροισμα κυρτών συναρτήσεων. Τελικά, λοιπόν κάθε κόμβος i έχει να λύσει το παρακάτω κυρτό πρόβλημα ελαχιστοποίησης:

$$\min_{\lambda_{ij}} \sum_{j=1, j\neq i}^{N} \lambda_{ij} \frac{1}{(1-\sum_{k\leq i}\lambda_{kj})(1-\sum_{k<i}\lambda_{kj})} \tag{4.5}$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{N} \lambda_{ij} = \lambda_i \qquad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} > 0 \qquad \forall i, j$$

Ένας τρόπος επίλυσης του παραπάνω προβλήματος είναι η εφαρμογή των συνθηκών ΚΚΤ. Ωστόσο, στη συνέχεια θα παρουσιάσουμε δύο πρωτόκολλα, που

βασίζονται στην μέθοδο του waterfilling και χαρακτηρίζονται από μικρή πολυπλοκότητα και ελάχιστη ανταλλαγή μηνυμάτων ανάμεσα στους χρήστες. Η μόνη γνώση που απαιτείται σε αυτή την προσέγγιση είναι η γνώση των μερικών παραγώγων της παραπάνω αντικειμενικής συνάρτησης, οι οποίες είναι

$$\frac{\partial F}{\partial \lambda_{ij}} = b_i \left(D_{ij} + \frac{\lambda_{ij}}{\left(1 - \sum_{k \le i} \lambda_{kj}\right)^2 \left(1 - \sum_{k<i} \lambda_{kj}\right)} \right) \tag{4.6}$$

με $D_{ij} = \frac{1}{\left(1 - \sum_{k \le i} \lambda_{kj}\right)\left(1 - \sum_{k<i} \lambda_{kj}\right)}$

Πρωτόκολλο 1

Όπως φαίνεται από την έκφραση (4.6), η μόνη πληροφορία που χρειάζεται ο κόμβος i για να υπολογίσει τις μερικές παραγώγους και να εκτελέσει τον αλγόριθμο είναι η ποσότητα $\sum_{k<i} \lambda_{kj}$ για κάθε εξυπηρετητή j, δηλαδή το άθροισμα των σπασιμάτων των κόμβων μεγαλύτερης προτεραιότητας σε κάθε κόμβο j. Επίσης, η καθυστέρηση του κόμβου i δεν εξαρτάται από τα σπασίματα των κόμβων μικρότερης προτεραιότητας.

Βάσει των παραπάνω παρατηρήσεων, το πρωτόκολλο μπορεί να έχει ως εξής: Αρχικά όλοι οι κόμβοι ανακοινώνουν στο δίκτυο τα βάρη τους b_i και επομένως κάθε κόμβος γνωρίζει την προτεραιότητα του σε σχέση με τους άλλους. Στη συνέχεια οι κόμβοι με σειρά μειούμενης προτεραιότητας (ή ισοδύναμα κατά μειούμενα b_i) εκτελούν τον παραπάνω αλγόριθμο. Ο κόμβος i λοιπόν δέχεται ένα διάνυσμα $\mathbf{Sum_i} = \left(\sum_{k<i} \lambda_{kj} : j = 1,...,N \right)$ από τον κόμβο i-1 και λύνει το παραπάνω πρόβλημα βελτιστοποίησης, αποφασίζει δηλαδή το διάνυσμα των σπασιμάτων του $\boldsymbol{\lambda_i} = \left(\lambda_{ij} : j = 1,...N, j \neq i \right)$, έτσι ώστε να μην παραβιάζεται η συνθήκη ευστάθειας του συστήματος σε κανέναν εξυπηρετητή. Ωστόσο, μέχρι αυτή την στιγμή έχουν επιλέξει τα σπασίματα τους μόνο οι κόμβοι μεγαλύτερης προτεραιότητας του i, οπότε η συνθήκη ευστάθειας είναι η $\sum_{k=1}^{i} \lambda_{kj} < 1$.Το γεγονός όμως αυτό έχει ως αποτέλεσμα, σε περιπτώσεις που $\sum_{i=1}^{N} \lambda_i \simeq \sum_{j=1}^{N} C_j$ να παρουσιάζεται το φαινόμενο ο κόμβος i να αδυνατεί να βρει ένα διάνυσμα $\boldsymbol{\lambda_i}$, ώστε να διατηρήσει το σύστημα σε ευστάθεια, με δεδομένες τις επιλογές των κόμβων μεγαλύτερης προτεραιότητας. Το φαινόμενο αυτό θα παρουσιαστεί αναλυτικότερα στη συνέχεια. Ακολούθως, εφόσον δεν παρουσιάζεται το πρόβλημα που μόλις αναφέραμε, προσθέτει τα δικά του σπασίματα στο διάνυσμα $\mathbf{Sum_i}$ και έτσι δημιουργεί το $\mathbf{Sum_{i+1}}$ το οποίο και αποστέλλει στον i+1 κόμβο.

Αν οι κόμβοι εκτελέσουν τον αλγόριθμο με την προαναφερθείσα σειρά, απαιτούνται μόλις N εκτελέσεις του αλγορίθμου, μία για κάθε κόμβο. Το αποτέλεσμα της παραπάνω διαδικασίας είναι ο εγωιστικός πίνακας σπασιμάτων $\boldsymbol{\Lambda_e}$ και το $\boldsymbol{D_e}=(D_{e,1},....,D_{e,N})$ είναι το βέλτιστο εφικτό διάνυσμα καθυστερήσεων του εγωιστικού αλγορίθμου.

Πρωτόκολλο 2

Ένα εναλλακτικό πρωτόκολλο θα μπορούσε να προϋποθέτει την πραγματοποίηση της ανανέωσης των σπασιμάτων κάθε κόμβου με τυχαία σειρά. Σε αυτή την περίπτωση ο κόμβος i εκτός της ποσότητας $\sum_{k<i}\lambda_{kj}$, χρειάζεται επιπλέον να γνωρίζει και την ποσότητα $\sum_{k=1}^{N}\lambda_{kj}$ για κάθε εξυπηρετητή j, ώστε να εξασφαλίζει την ευστάθεια του συστήματος. Επίσης, η καθυστέρηση του κόμβου i δεν εξαρτάται άμεσα από τα σπασίματα των κόμβων μικρότερης προτεραιότητας, παρά μόνο με την έννοια ότι είναι πιθανό να περιορίζεται στις επιλογές του από την συνθήκη ευστάθειας του συστήματος.

Βάσει των παραπάνω παρατηρήσεων, το πρωτόκολλο μπορεί να έχει ως εξής: Το σύστημα ξεκινά από μία αρχική κατάσταση όπου κάθε κόμβος θεωρεί ένα τυχαίο πίνακα σπασιμάτων του συστήματος $\boldsymbol{\Lambda^0}$. Σε μια τυχαία στιγμή ένας οποιοσδήποτε κόμβος i επιλέγει να εκτελέσει τον παραπάνω αλγόριθμο. Ο κόμβος i λοιπόν έχει τον πίνακα $\boldsymbol{\Lambda}$ στη διάθεση του και λύνει το παραπάνω πρόβλημα βελτιστοποίησης, αποφασίζει δηλαδή το διάνυσμα των σπασιμάτων του $\boldsymbol{\lambda_i}=\left(\lambda_{ij}: j=1,...N, j\neq i\right)$, εξασφαλίζοντας ταυτόχρονα ότι δεν παραβιάζεται η συνθήκη ευστάθειας του συστήματος σε κανέναν εξυπηρετητή. Ακολούθως, ανακοινώνει σε όλους τους άλλους κόμβους τα νέα σπασίματα που επέλεξε, δηλαδή το διάνυσμα $\boldsymbol{\lambda_i}$, οι οποίοι βάσει αυτού ενημερώνουν τον πίνακα $\boldsymbol{\Lambda}$.

Το πρωτόκολλο που μόλις περιγράψαμε, όπως θα δούμε και στη συνέχεια αναλυτικότερα, μετά από έναν αριθμό εκτελέσεων του αλγορίθμου από κάθε κόμβο συγκλίνει, δηλαδή κανείς κόμβος πλέον δεν τροποποιεί τα σπασίματά του. Το αποτέλεσμα της παραπάνω διαδικασίας είναι ο εγωιστικός πίνακας σπασιμάτων $\boldsymbol{\Lambda_\varepsilon}$ και το $\boldsymbol{D_e}=(D_{e,1},....,D_{e,N})$ είναι το βέλτιστο εφικτό διάνυσμα καθυστερήσεων του εγωιστικού αλγορίθμου για το πρωτόκολλο αυτό.

Προσομοίωση

Για την καλύτερη κατανόηση της συμπεριφοράς και την αξιολόγηση της απόδοσης των παραπάνω αλγορίθμων προχωρήσαμε στην υλοποίησή τους και στην προσομοίωση ενός δικτύου βάσει του μοντέλου που περιγράψαμε προηγουμένως. Το σύνολο των προσομοιώσεων πραγματοποιήθηκε σε περιβάλλον MATLAB και γενικά, εκτός αν αναφέρεται διαφορετικά, το δίκτυο που μοντελοποιούμε αποτελείται από N=4 κόμβους με διάνυσμα βαρών το **b**=[4 3 2 1]. Επίσης, όλοι οι κόμβοι έχουν τον

ίδιο ρυθμό παραγωγής αιτήσεων λ_i και το βήμα της τεχνικής waterfilling έχει τιμή $\varepsilon = 10^{-3}$. Στη συνέχεια παρουσιάζουμε κάποια από τα αποτελέσματα που προέκυψαν.

Πρωτόκολλο 1

Απόδοση αλγορίθμου

Όπως έχουμε ήδη επισημάνει στο ορισμό του πρωτοκόλλου χρειάζεται μόλις μία εκτέλεση του αλγορίθμου για την εύρεση του βέλτιστου πίνακα σπασιμάτων. Ωστόσο, για μικρό αριθμό κόμβων και για μεγάλο φόρτο αιτήσεων στο σύστημα, παρατηρείται το φαινόμενο ο τελευταίος από τους κόμβους χαμηλής προτεραιότητας να αδυνατεί να βρει ένα διάνυσμα $\boldsymbol{\lambda}_i$ που να ικανοποιεί την συνθήκη ευστάθειας του συστήματος. Ενδεικτικά, θα αναφέρουμε ένα παράδειγμα όπου εμφανίζεται το παραπάνω φαινόμενο.

Αν έχουμε $\lambda_i = 0.95$ για κάθε κόμβο ο αλγόριθμος δίνει τον παρακάτω πίνακα σπασιμάτων.

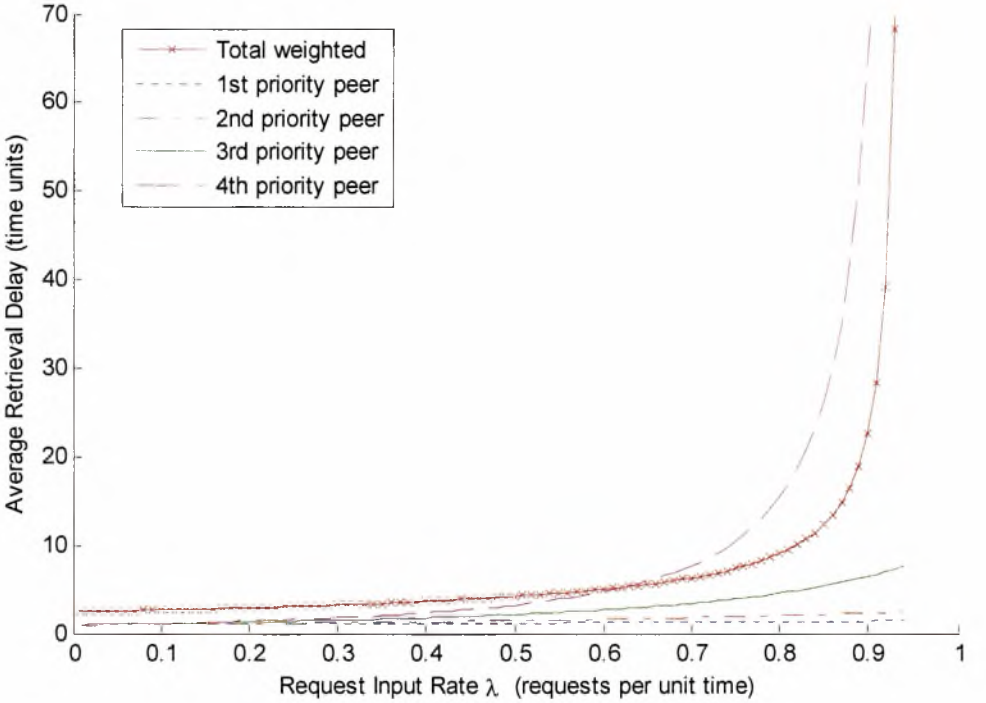
Server / Client	1	2	3	4
1	0	0.3166	0.3167	0.3167
2	0.5278	0	0.2110	0.2112
3	0.2463	0.4575	0	0.2461
4	0.2258	0.2258	-1	0

Όπως, καταδεικνύει και ο παραπάνω πίνακας ο πρώτος χρήστης (αυτός με την μεγαλύτερη προτεραιότητα) σπάει εξίσου τις αιτήσεις του. Ακολούθως, καθώς πρόκειται για ένα πρόβλημα εξισορρόπησης φόρτου (load balancing) ο δεύτερος κόμβος στέλνει 0.3166 στον πρώτο και το υπόλοιπο (0.95-0.3166) το διαμοιράζει εξίσου σε όλους. Ομοίως και ο τρίτος. Όμως, ο τέταρτος πλέον δεν έχει αρκετή διαθέσιμη χωρητικότητα με αποτέλεσμα να αδυνατεί να διατηρήσει την συνθήκη ευστάθειας (το οποίο δηλώνεται με το -1) στον κόμβο 3. Το φαινόμενο αυτό παρατηρείται λόγω της αδιαφορίας που επιδεικνύουν οι κόμβοι μεγαλύτερης προτεραιότητας για τους υπολοίπους, λόγω του ότι οι δεύτεροι δεν επηρεάζουν την καθυστέρηση τους. Έτσι λοιπόν στο παράδειγμά μας, οι τρεις πρώτοι κόμβοι κάνουν τις επιλογές τους με μοναδικό κριτήριο την καθυστέρηση που θα αντιμετωπίσουν και αγνοούν την ύπαρξη των κόμβων χαμηλότερης προτεραιότητας. Αυτό έχει ως αποτέλεσμα τελικά να παραμένει αδιάθετη χωρητικότητα στον εξυπηρετητή 4. Το φαινόμενο αυτό γενικά εμφανίζεται λιγότερες φορές καθώς αυξάνουμε τον αριθμό των κόμβων του συστήματος. Αυτό οφείλεται στο ότι επιτυγχάνεται καλύτερος διαμοιρασμός του φόρτου και παραμένει αδιάθετο μικρότερο ποσοστό χωρητικότητας στον τελευταίο εξυπηρετητή.

Στο γράφημα φαίνεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικός βεβαρημένος μέσος χρόνος ανάκτησης του συστήματος για διάφορες τιμές του ρυθμού παραγωγής αιτήσεων λ_i. Σημειώνουμε, ότι για να είναι πιο ευδιάκριτες οι γραφικές παραστάσεις, ιδιαίτερα σε μικρές τιμές των λ_i, στον κατακόρυφο άξονα εμφανίζεται μόνο το διάστημα [0-70] αν και οι γραφικές παραστάσεις της καθυστέρησης για τον τέταρτο κόμβο και του συνολικού βεβαρημένου αθροίσματος εκτείνονται και εκτός αυτού του διαστήματος.

Επίσης, η έλλειψη σημείων για ορισμένα λ_i υποδηλώνει την αδυναμία εύρεσης εφικτής λύσης. Στο συγκεκριμένο παράδειγμα βλέπουμε ότι για τιμές $\lambda_i \geq 0.95$ ο αλγόριθμος δεν μπορεί να βρει λύση που να ικανοποιεί την συνθήκη ευστάθειας.

OBJECTIVE 1: REQUEST THROUGHPUT - DELAY PERFORMANCE FOR EGOTISTIC APPROACH (P1)



Το πρώτο που παρατηρεί κανείς στο γράφημα είναι ότι καθώς αυξάνεται ο ρυθμός παραγωγής αιτήσεων, η καθυστέρηση αυξάνεται ιδιαίτερα. Επίσης είναι εμφανές ότι γενικά ο ρυθμός αύξησης της καθυστέρησης ενός κόμβου εξαρτάται από την προτεραιότητά του. Έτσι λοιπόν ο κόμβος 1 πού έχει απόλυτη προτεραιότητα δεν επηρεάζεται ιδιαίτερα από την αύξηση του λ_i, ενώ ο κόμβος 4 βλέπει την καθυστέρηση του να αυξάνει με ταχύτατο ρυθμό. Αιτία αυτού του φαινομένου είναι ότι ουσιαστικά ο κόμβος 1 έχοντας απόλυτη προτεραιότητα επηρεάζεται μόνο από την αύξηση του δικού του ρυθμού παραγωγής αιτήσεων, ενώ ο κόμβος 4 από την αύξηση όλων των λ_i. Γενικά κάθε κόμβος επηρεάζεται μόνο από την αύξηση των ρυθμών παραγωγής των κόμβων μεγαλύτερης προτεραιότητας.

Επίσης, η συνολική καθυστέρηση στο σύστημα, όντας ο βεβαρημένος μέσος όρος των καθυστερήσεων, με βάρη μεγαλύτερα της μονάδας, θα περιμέναμε να βρίσκεται πάνω από τις επιμέρους καθυστερήσεις των κόμβων. Αυτό γενικά ισχύει, όχι όμως και για μεγάλες τιμές του λ_i. Σε αυτές τις περιπτώσεις η τιμή της συνολικής καθυστέρησης είναι μικρότερη από αυτή του κόμβου 4 λόγω της ραγδαίας αύξησης που περιγράψαμε παραπάνω.

Τέλος, επιβεβαιώνεται ότι οι κόμβοι μεγαλύτερης προτεραιότητας έχουν μικρότερη καθυστέρηση. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.9$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα

D_1	D_2	D_3	D_4	Total Delay
1.429	2.381	6.468	64.46	22.56

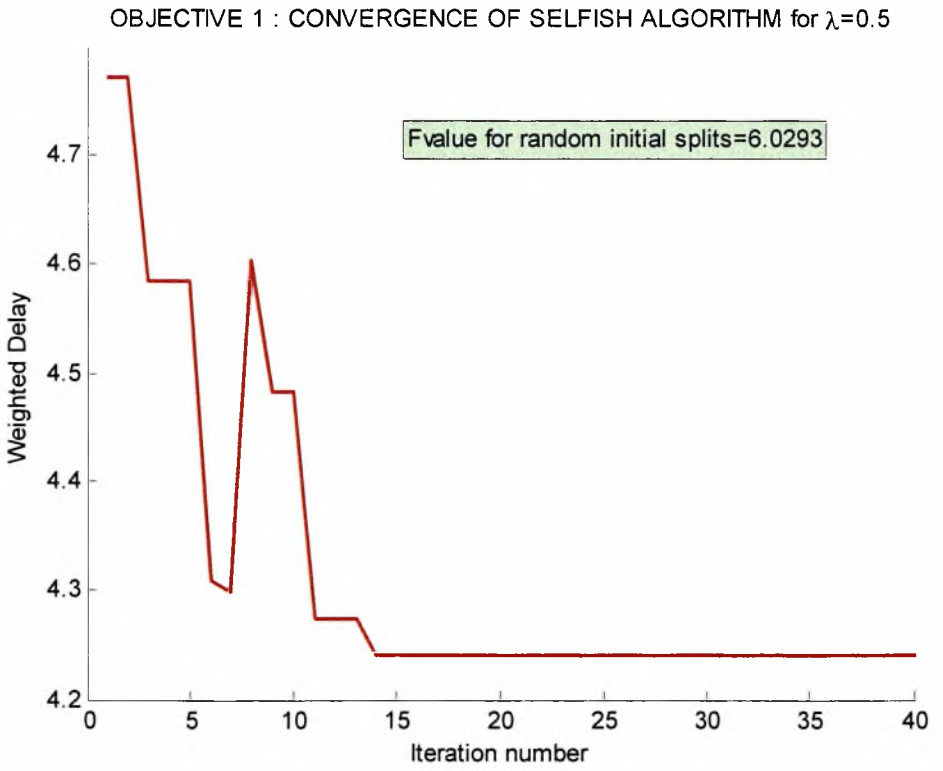
Πρωτόκολλο 2

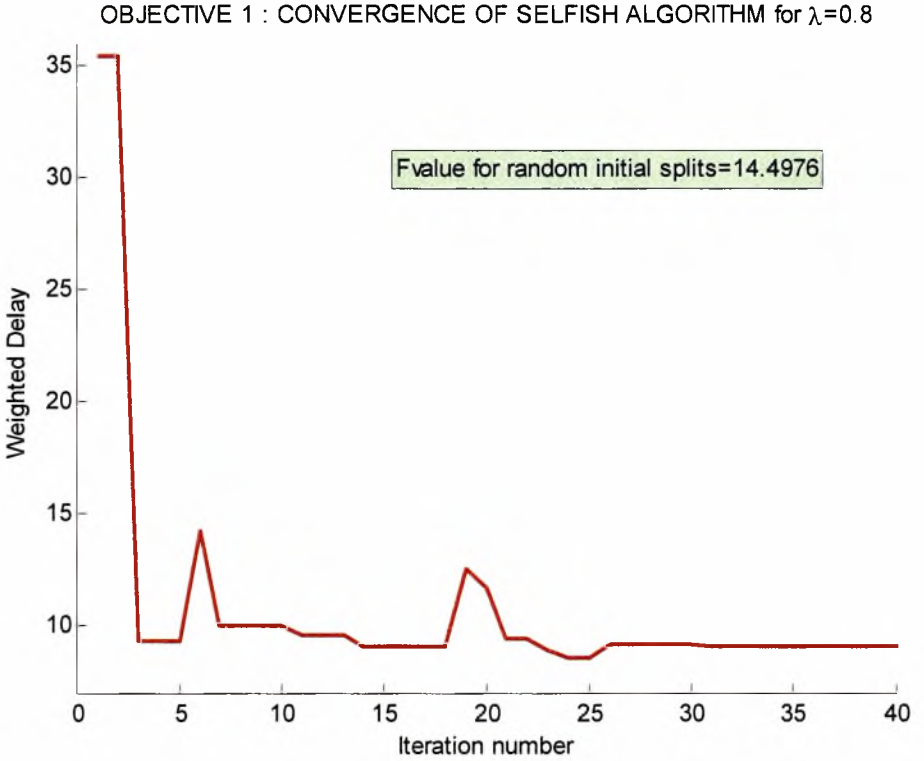
I. Σύγκλιση αλγορίθμου

Η δυνατότητα αυτή που αναφέραμε παραπάνω δεν μας παρέχεται χωρίς κόστος. Το κόστος λοιπόν είναι ότι πλέον δεν αρκεί μια εκτέλεση του αλγορίθμου για κάθε κόμβο, αλλά η διαδικασία αυτή πρέπει να επαναλαμβάνεται. Ωστόσο, μετά από έναν αριθμό εκτελέσεων του αλγορίθμου από κάθε κόμβο επιτυγχάνεται σύγκλιση, δηλαδή κανείς κόμβος πλέον δεν τροποποιεί τα σπασίματά του. Είναι αυτονόητο ότι καθώς ο αριθμός των κόμβων του συστήματος αυξάνεται απαιτούνται περισσότερες επαναλήψεις ώστε να επιτευχθεί η σύγκλιση.

Στη συνέχεια, παρουσιάζουμε την ταχύτητα σύγκλισης του αλγορίθμου για δύο διαφορετικές περιπτώσεις ρυθμών αφίξεων λ για να δούμε πως συμπεριφέρεται ο αλγόριθμος για διαφορετικό φόρτο αιτήσεων. Συγκεκριμένα, απεικονίζουμε την τιμή της αντικειμενικής συνάρτησης μετά από κάθε εκτέλεση του αλγορίθμου από ένα κόμβο.

Σε αυτό το σημείο είναι απαραίτητο να αναφέρουμε ότι εξαιτίας και της τυχαίας επιλογής του αρχικού πίνακα σπασιμάτων Λ^0, ο αριθμός των επαναλήψεων που απαιτούνται για την επίτευξη της σύγκλισης δεν είναι σταθερός. Ωστόσο, στα παρακάτω γραφήματα παρουσιάζουμε μια μέση περίπτωση όπου μπορεί κανείς να βγάλει ασφαλή συμπεράσματα για την ταχύτητα σύγκλισης του αλγορίθμου.





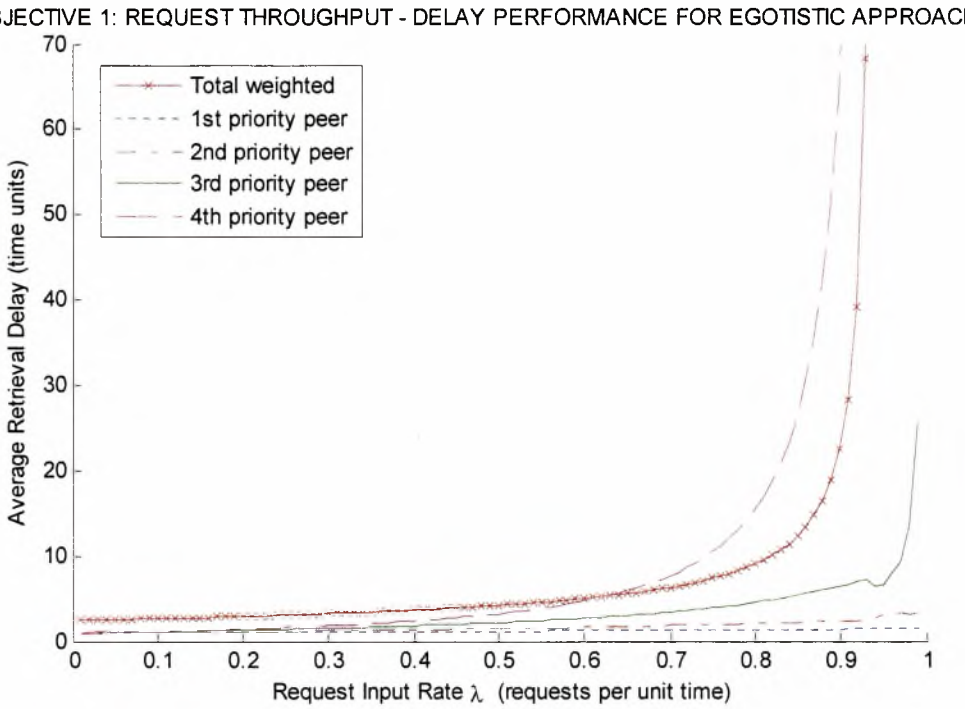
Όπως φαίνεται και από τα γραφήματα η μέση καθυστέρηση του συστήματος δεν μειώνεται πάντα μεταξύ δύο επαναλήψεων. Αυτό οφείλεται στο γεγονός ότι ο κάθε χρήστης προσπαθεί να ελαχιστοποιήσει την δικιά του καθυστέρηση αδιαφορώντας για την καθυστέρηση των άλλων. Παρ' όλα αυτά τελικά μετά από έναν αριθμό επαναλήψεων ο αλγόριθμος συγκλίνει και μάλιστα όπως είδαμε και προηγουμένως συγκλίνει στη λύση που δίνει και το πρωτόκολλο 1. Πιο συγκεκριμένα, οι κορυφές που εμφανίζονται στα γραφήματα οφείλονται στην εκτέλεση του αλγορίθμου από κάποιον κόμβο μεγάλης προτεραιότητας. Σε αυτές τις περιπτώσεις ο συγκεκριμένος κόμβος επιλέγει το καλύτερο διάνυσμα σπασιμάτων με κριτήριο τη δικιά του καθυστέρηση και μόνο. Αυτό, όμως μπορεί να έχει ως αποτέλεσμα να μην γίνεται καλός διαμοιρασμός του φόρτου στο σύστημα και σε κάποιον εξυπηρετητή να έχουμε λειτουργία στα όρια της περιοχής ευστάθειας. Συνεπώς η καθυστέρηση των κόμβων χαμηλής προτεραιότητας αυξάνεται πολύ συμπαρασύροντας και την συνολική καθυστέρηση στο σύστημα. Ενδεικτικό του γεγονότος αυτού είναι και το ότι στο δεύτερο γράφημα, όπου ο φόρτος είναι μεγαλύτερος παρουσιάζονται μεγαλύτερες διακυμάνσεις.

Λόγω των παραπάνω, στην πρώτη περίπτωση, όπου λ=0.5, ο αλγόριθμος χρειάζεται μόλις 15 επαναλήψεις για να συγκλίνει ενώ λ=0.8 η σύγκλιση επιτυγχάνεται γύρω στην τριακοστή επανάληψη. Επίσης, στο πλαίσιο εμφανίζεται η καθυστέρηση του συστήματος για τον τυχαίο αρχικό πίνακα σπασιμάτων Λ^0. Η τιμή αυτή υποδεικνύει την βελτίωση που επιτυγχάνεται με την χρήση του αλγορίθμου για την εύρεση των σπασιμάτων σε σχέση με μια τυχαία επιλογή σπασιμάτων.

II. Απόδοση αλγορίθμου

Το πρωτόκολλο αυτό μας δίνει την δυνατότητα η ανανέωση των σπασιμάτων κάθε κόμβου να πραγματοποιείται με τυχαία σειρά, ενώ ταυτόχρονα λόγω της ύπαρξης του αρχικού τυχαίου πίνακα σπασιμάτων Λ^0 και της συνθήκης ευστάθειας εξασφαλίζει ότι το σύστημα θα εντοπίσει μια εφικτή λύση σε κάθε περίπτωση όπου $\sum_{i=1}^{N} \lambda_i < \sum_{j=1}^{N} \mu_j$.

Όπως και για το προηγούμενο πρωτόκολλο λοιπόν ,στο παρακάτω γράφημα παρουσιάζεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικός βεβαρημένος μέσος χρόνος ανάκτησης του συστήματος για διάφορες τιμές του ρυθμού παραγωγής αιτήσεων λ_i. Και σε αυτή την περίπτωση πρέπει να αναφέρουμε, ότι για να είναι πιο ευδιάκριτες οι γραφικές παραστάσεις, ιδιαίτερα σε μικρές τιμές των λ_i, στον κατακόρυφο άξονα εμφανίζεται μόνο το διάστημα [0-70] αν και οι γραφικές παραστάσεις της καθυστέρησης για τον τέταρτο κόμβο και του συνολικού βεβαρημένου αθροίσματος εκτείνονται και εκτός αυτού του διαστήματος.



Πέραν των παρατηρήσεων που επισημάναμε στο πρωτόκολλο 1, οι οποίες ισχύουν και στην περίπτωση αυτού του πρωτοκόλλου, παρατηρούμε ότι πλέον είναι δυνατή η εύρεση λύσης και για τις περιπτώσεις όπου $\lambda_i \geq 0.95$. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.9$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα

D_1	D_2	D_3	D_4	**Total Delay**
1.429	2.381	6.468	64.46	22.56

Συγκρίνοντας με τον πίνακα του πρωτοκόλλου 1 διαπιστώνουμε ότι η απόδοση των δύο πρωτοκόλλων είναι ταυτόσημη για τις τιμές των λ_i που βρίσκουν εφικτή

λύση. Αυτό ισχύει γενικά, ανεξάρτητα από την σειρά με την οποία οι κόμβοι εκτελούν τις ανανεώσεις των σπασιμάτων τους (έχουμε ήδη αναφέρει ότι η σειρά είναι τυχαία) και επίσης ανεξάρτητα από τον αρχικό πίνακα σπασιμάτων Λ^0.

Η διαφορά έγκειται στο ότι το δεύτερο πρωτόκολλο βρίσκει μια εφικτή λύση και για τις άλλες περιπτώσεις έστω και αν αυτή χαρακτηρίζεται από πολύ μεγάλη καθυστέρηση.

Ειδικότερα για $\lambda_i = 0.95$ ο αλγόριθμος επιστρέφει τώρα τον παρακάτω πίνακα σπασιμάτων

Server / Client	1	2	3	4
1	0	0.3166	0.3163	0.3172
2	0.3884	0	0.2821	0.2796
3	0.3399	0.4037	0	0.2064
4	0.2708	0.2787	0.4005	0

Όπως, καταδεικνύει και ο παραπάνω πίνακας η εξισορρόπηση φόρτου δεν έχει γίνει με ιδιαίτερα καλό τρόπο, όμως, αντίθετα με προηγουμένως, κατάφερε τουλάχιστον να εξασφαλιστεί η ευστάθεια του συστήματος.

Packet – oriented Προσομοίωση Δικτύου

Οι αλγόριθμοι που παρουσιάσαμε μέχρι τώρα έχουν σαν κοινό χαρακτηριστικό ότι οι προτεραιότητες εξυπηρέτησης είναι προκαθορισμένες, ίδιες για όλους τους κόμβους και σταθερές σε όλη τη διάρκεια της προσομοίωσης. Στη συνέχεια θα παρουσιάσουμε έναν ευρεστικό αλγόριθμο ελαχιστοποίησης της καθυστέρησης όπου οι προτεραιότητες δεν είναι καθολικές, αλλά διαμορφώνονται δυναμικά με βάση το ρυθμό εξυπηρέτησης κάθε κόμβου, ο οποίος μπορεί να είναι διαφορετικός για κάθε εξυπηρετητή. Ο αλγόριθμος αυτός θα χρησιμοποιηθεί για να διαπιστώσουμε την επίδραση της εγωιστικής συμπεριφοράς στον καθορισμό των προτεραιοτήτων.

Το μοντέλο της προσομοίωσης

Το μοντέλο που προσομοιώνουμε αποτελείται από ένα δίκτυο τριών κόμβων. Καθένας από τους κόμβους λειτουργεί ως *πελάτης (client)* και ως *εξυπηρετητής (server)* σε ένα δίκτυο διαμοιρασμού αρχείων. Ως, πελάτης, κάθε κόμβος ζητά, σε τυχαίες χρονικές στιγμές, αρχεία από τους άλλους δύο, ενώ ως εξυπηρετητής ικανοποιεί τις αιτήσεις των άλλων κόμβων, που φτάνουν σε αυτόν, για αρχεία. Κάθε κόμβος μπορεί να έχει διαφορετικό ρυθμό παραγωγής αιτήσεων και ρυθμό εξυπηρέτησης από τους υπόλοιπους φροντίζοντας όμως πάντα να υπάρχει ευστάθεια σε κάθε κόμβο και επομένως και συνολικά στο δίκτυο, δηλαδή σε κανένα κόμβο να μην υπάρχει συνολικός ρυθμός άφιξης αιτήσεων μεγαλύτερος από το αντίστοιχο ρυθμό εξυπηρέτησης. Επίσης, είναι απαραίτητο να αναφέρουμε ότι πρόκειται για ένα *non-preemptive μοντέλο εξυπηρέτησης* δηλαδή όταν μια αίτηση αρχίσει να εκτελείται δεν πρόκειται να διακοπεί ακόμη και αν έρθει στην ουρά μια αίτηση με μεγαλύτερη προτεραιότητα.

Συμπεριφορά κόμβου ως πελάτης – client

Ο κόμβος i γεννά αιτήσεις που απευθύνονται στους άλλους κόμβους με ρυθμό λ_i. Οι χρονικές στιγμές της γέννησης των αιτήσεων αυτών ακολουθούν κατανομή *Poisson* και επομένως τα διαστήματα μεταξύ των αιτήσεων ακολουθούν την εκθετική κατανομή. Ο συνολικός ρυθμός παραγωγής αιτήσεων λ_i για τον κόμβο i, χωρίζεται σε δύο επιμέρους ρυθμούς, τον ρυθμό με τον οποίο θα στέλνονται οι αιτήσεις στον ένα γείτονα, και στον ρυθμό που θα στέλνονται στον άλλο. Για παράδειγμα, ο κόμβος 1 παράγει αιτήσεις με ρυθμό λ_1, από τις οποίες κάποιες, με ρυθμό λ_{12}, στέλνονται στον κόμβο 2 και με λ_{13} στον κόμβο 3. Δηλαδή σε κάθε στιγμή για τον κόμβο 1 ισχύει: $\lambda_1 = \lambda_{12} + \lambda_{13}$. Ο τρόπος με τον οποίο γίνεται το «σπάσιμο» του ρυθμού παραγωγής των αιτήσεων και συνεπώς η απόφαση για τον κόμβο στον οποίο θα σταλεί η κάθε αίτηση θα εξηγηθεί στη συνέχεια.

Μετά από τη γέννηση μιας αίτησης, ο κόμβος πρέπει να αποφασίσει σε ποιον θα στείλει την αίτηση. Ο κόμβος αποφασίζει τυχαία τον κόμβο στον οποίο θα στείλει την αίτηση με πιθανότητα όμως μεγαλύτερη να την στείλει στον κόμβο για τον οποίο έχει καλύτερη γνώμη. Η άποψη ενός κόμβου για τους γείτονές του διαμορφώνεται με βάση τη συμπεριφορά τους στα αιτήματά του στη διάρκεια των προηγούμενων εποχών. Έτσι, ο γείτονας που επέφερε μικρότερη μέση καθυστέρηση στην

εξυπηρέτηση των αιτήσεών του θα είναι και αυτός για τον οποίο έχει καλύτερη γνώμη.

Συμπεριφορά κόμβου ως εξυπηρετητής - server

Κάθε κόμβος χαρακτηρίζεται από το ρυθμό εξυπηρέτησης, ο οποίος στο πρόβλημα που μελετάμε αντικατοπτρίζει τη χωρητικότητα της γραμμής με την οποία συνδέεται στο δίκτυο. Επιπλέον, σε κάθε κόμβο φτάνουν αιτήσεις που προέρχονται από τους δύο γείτονές του. Το γεγονός αυτό καθιστά απαραίτητη τη χρησιμοποίηση κάποιου αλγορίθμου εξυπηρέτησης των αιτήσεων. Υπάρχουν διάφοροι τρόποι εξυπηρέτησης, καθένας από τους οποίους δίνει διαφορετικά αποτελέσματα όσον αφορά την καθυστέρηση κάθε κόμβου ξεχωριστά και τη συνολική καθυστέρηση του συστήματος. Οι μέθοδοι καθορισμού της σειράς εξυπηρέτησης που θα εξετάσουμε είναι οι: *First Come First Served*, *Shortest Job First* καθώς και μία μέθοδος που χρησιμοποιεί ένα δυναμικό τρόπο καθορισμού των προτεραιοτήτων εξυπηρέτησης. Όλες αυτές οι μέθοδοι θα εξηγηθούν στη συνέχεια.

Η δομή της προσομοίωσης

Η προσομοίωση είναι οργανωμένη σε *εποχές (epochs)*. Σε κάθε μία τέτοια εποχή προσομοιώνουμε το δίκτυο με τους διάφορους τρόπους εξυπηρέτησης για ένα συγκεκριμένο χρονικό διάστημα. Κατά τη διάρκεια μια εποχής, κάθε κόμβος, ανάλογα με το ρυθμό λ που έχει, παράγει έναν αριθμό αιτήσεων.

Ως πελάτης, κάθε κόμβος αποφασίζει σε ποιον από τους δύο γείτονες θα στείλει κάθε αίτηση ανάλογα με τη γνώμη του για αυτούς. Η γνώμη κάθε κόμβου για τους άλλους διαμορφώνεται βάσει της μέσης καθυστέρησης που είχαν οι αιτήσεις του σε αυτούς στο σύνολο των προηγούμενων εποχών. Συγκεκριμένα, αν $wait(1,2)$ και $wait(1,3)$ η μέση καθυστέρηση του κόμβου 1 στους 2 και 3 αντίστοιχα, η γνώμη του διαμορφώνεται βάσει των παρακάτω τύπων:

$$reputation(1,2) = \frac{wait(1,3)}{wait(1,2) + wait(1,3)} \tag{4.7}$$

και

$$reputation(1,3) = \frac{wait(1,2)}{wait(1,2) + wait(1,3)} \tag{4.8}$$

Επίσης, κάθε κόμβος εξυπηρετεί τις αιτήσεις που καταφθάνουν σε αυτόν με κάποια μέθοδο από τις παρακάτω.

First Come – First Served

Κάθε αίτηση εξυπηρετείται με βάση τη σειρά άφιξης στην ουρά. Η αίτηση που ήρθε πρώτη εξυπηρετείται και πρώτη. Αυτή η μέθοδος δίνει περισσότερο έμφαση στη δικαιοσύνη εξυπηρέτησης και λιγότερο στην αποδοτική εξυπηρέτηση των αιτήσεων. Αυτό συμβαίνει γιατί αιτήματα που απαιτούν πολύ χρόνο για να ικανοποιηθούν θα

καθυστερούν υπερβολικά τις μικρότερες, ειδικά όταν οι χρόνοι εξυπηρέτησης των αιτήσεων έχουν μεγάλη διασπορά. Επιπλέον, αποφεύγεται ο κίνδυνος λιμοκτονίας.

Shortest Job First

Κάθε χρονική στιγμή εξυπηρετείται η αίτηση με το μικρότερο χρόνο εξυπηρέτησης. Με τον τρόπο αυτό οι μικρότερες αιτήσεις παραγκωνίζουν τις μεγαλύτερες με κίνδυνο να υπάρξει λιμοκτονία. Ωστόσο, οι μικρότερες αιτήσεις δεν είναι υποχρεωμένες να περιμένουν πολύ χρόνο πίσω από άλλες με πολύ μεγάλο χρόνο εξυπηρέτησης. Η μέθοδος αυτή, όπως είναι εμφανές, δίνει έμφαση στην αποδοτική ικανοποίηση των αιτημάτων και έχει αποδειχθεί ότι συντελεί στη ελαχιστοποίηση του μέσου χρόνο αναμονής.

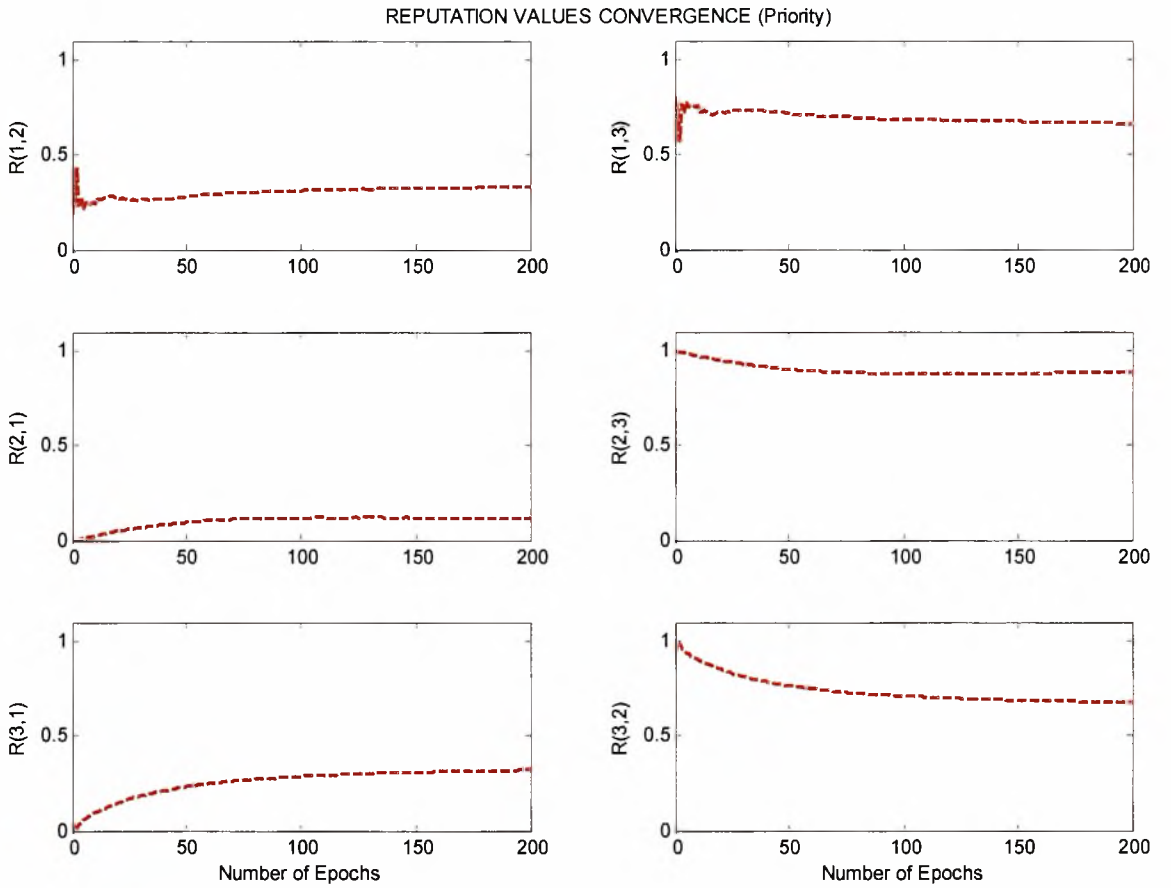
Μέθοδος εξυπηρέτησης με δυναμικό καθορισμό προτεραιοτήτων

Πρόκειται για μια μέθοδο εξυπηρέτησης όπου κάθε κόμβος έχει διαφορετική προτεραιότητα. Οι προτεραιότητες καθορίζονται στο τέλος κάθε εποχής, βάσει της τιμής reputation. Συγκεκριμένα, προτεραιότητα δίνεται στον κόμβο με το μεγαλύτερο reputation. Αυτός είναι και ένας τρόπος επιβράβευσης του γείτονα για την καλύτερη εξυπηρέτηση που του προσέφερε αφού έτσι θα εξυπηρετούνται με πολύ μικρή καθυστέρηση οι αιτήσεις του. Αυτή βέβαια είναι και μια αιτία δημιουργίας *Συμμαχιών (coalitions)* μεταξύ κάποιων κόμβων και εις βάρος κάποιων άλλων, φαινόμενο που θα αναλύσουμε και παρακάτω.

Προσομοίωση

I. Σύγκλιση αλγορίθμου

Όπως αναφέραμε και στον ορισμό του ευρεστικού αυτού αλγορίθμου ο κάθε κόμβος επιλέγει τα σπασίματά του βάσει της καθυστέρησης που αντιμετώπισε κατά την εξυπηρέτηση του από τους άλλους κόμβους τις προηγούμενες εποχές. Έτσι λοιπόν στο παρακάτω γράφημα παρουσιάζουμε την σύγκλιση των τιμών reputation για το δίκτυο των τριών κόμβων με μ=[0.5 1 1.5].



Παρατηρούμε ότι μετά από ένα αριθμό από εποχές οι τιμές reputation σταθεροποιούνται για όλους τους κόμβους. Ειδικότερα μπορούμε να δούμε ότι κάθε κόμβος στέλνει το μεγαλύτερο μέρος των αιτήσεων του στον κόμβο με τον μεγαλύτερο ρυθμό εξυπηρέτησης. Ενδεικτικά παραθέτουμε στον παρακάτω πίνακα τις τιμές του Reputation όταν έχει επιτευχθεί η σύγκλιση

Client \ Server	1	2	3
1	0	0.3361	0.6639
2	0.1125	0	0.8875
3	0.3221	0.6779	0

Όπως παρατηρούμε και στον παραπάνω πίνακα οι κόμβοι 2 και 3 είναι αυτοί που δέχονται το μεγαλύτερο μέρος των αιτήσεων και επίσης ο ένας δίνει στον άλλο μεγάλη τιμή reputation και συνεπώς και προτεραιότητα. Αυτό είναι ενδεικτικό της δημιουργίας συνέργιας ανάμεσα στους δύο κόμβους γεγονός που θα περιγράψουμε αναλυτικότερα και στη συνέχεια.

II. Απόδοση αλγορίθμου

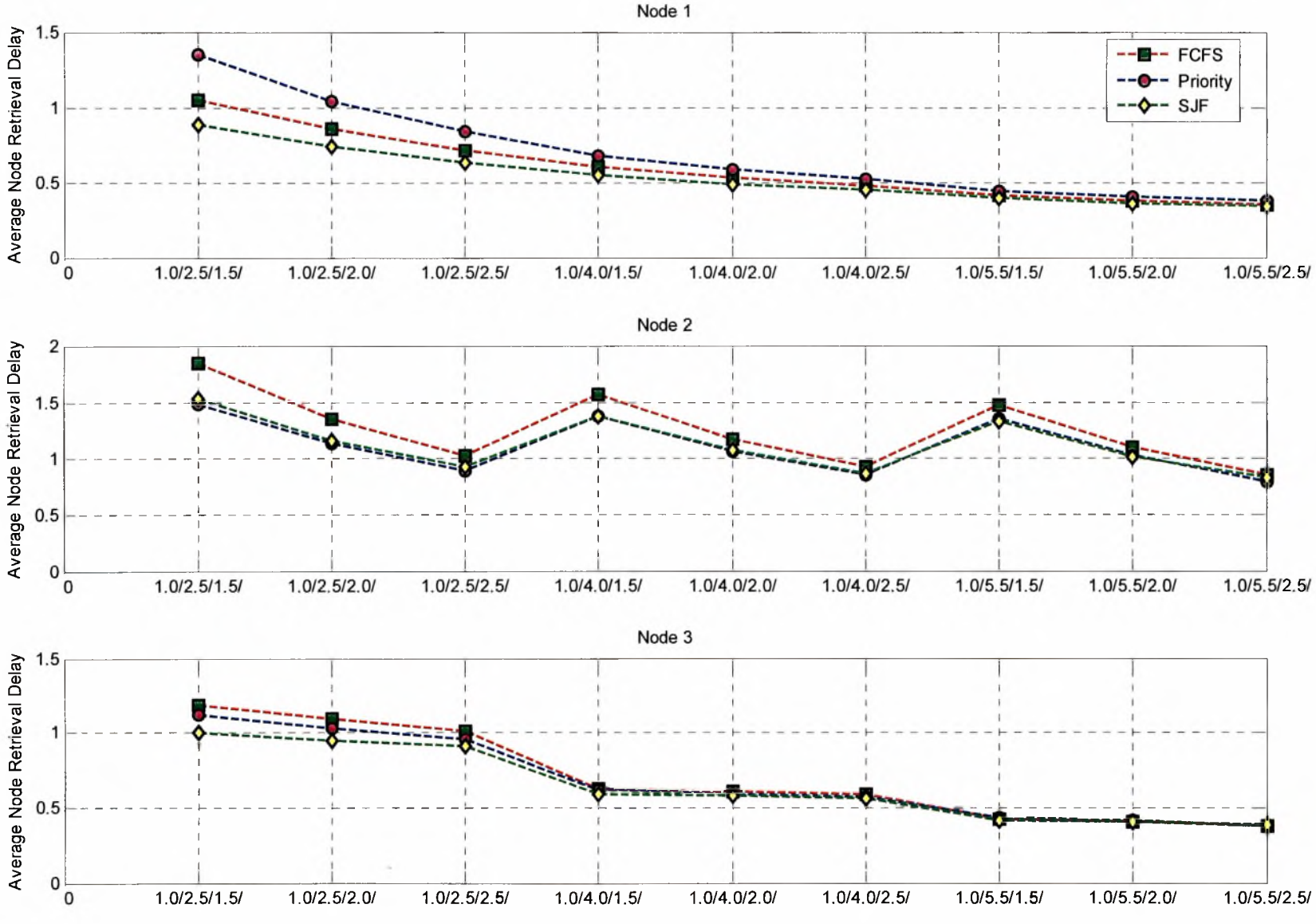
Για τη μέτρηση της απόδοσης του αλγορίθμου δυναμικών προτεραιοτήτων προσομοιώσαμε το σύστημα των τριών κόμβων για διάφορα διανύσματα C και συγκρίναμε τα αποτελέσματά του με εκείνα των μεθόδων εξυπηρέτησης *First Come First Served* και *Shortest Job First*. Εξαιτίας της διαφορετικής απόδοσης του αλγορίθμου για κάθε κόμβο λόγω της εμφάνισης συμμαχιών, παρουσιάζουμε παρακάτω τα διαγράμματα απόδοσης για κάθε κόμβο του συστήματος ξεχωριστά. Κατά τη διάρκεια της προσομοίωσης κρατήσαμε το ρυθμό εξυπηρέτησης $C(1)$ σταθερό και πάντα μικρότερο από τους άλλους δύο ρυθμούς ώστε να παρατηρήσουμε την εμφάνιση συμμαχίας μεταξύ των κόμβων 2 και 3.

Η πρώτη παρατήρηση που μπορούμε να κάνουμε αφορά τις κλίσεις των γραφικών, οι οποίες έχουν πτωτική πορεία εξαιτίας της συνεχόμενης αύξησης του συνολικού ρυθμού εξυπηρέτησης από πείραμα σε πείραμα. Όσον αφορά τον κόμβο 1, παρατηρούμε ότι σε κάθε περίπτωση ο αλγόριθμος προτεραιοτήτων του προκαλεί μεγαλύτερη καθυστέρηση σε σχέση με της άλλες μεθόδους, φαινόμενο που είναι απολύτως φυσιολογικό αφού ο 1 δε συμμετέχει σε καμία συμμαχία και επομένως τα αιτήματά του δεν έχουν προτεραιότητα σε καμία ουρά. Επιπλέον, επιβεβαιώνουμε ότι ο αλγόριθμος SJF έχει καλύτερα αποτελέσματα από τους άλλους δύο για τον κόμβο 1, αφού, όπως γνωρίζουμε από τη θεωρεία, ο καλύτερος τρόπος εξυπηρέτησης, όταν γνωρίζουμε το χρόνος εκτέλεσης των αιτημάτων, είναι κατά αύξοντα χρόνο εξυπηρέτησης.

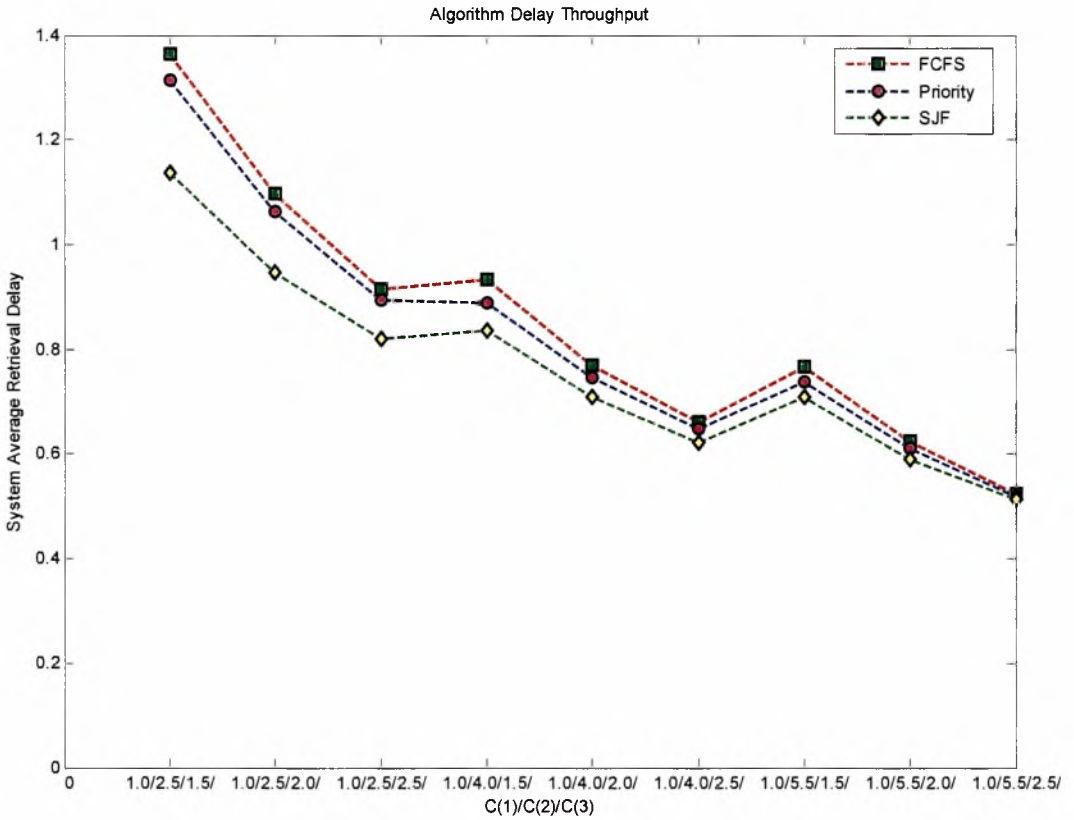
Για τους άλλους δύο κόμβους, όμως, τα συμπεράσματα είναι λίγο διαφορετικά. Οι κόμβοι αυτοί έχουν δημιουργήσει συμμαχία σε όλα τα πειράματα με αποτέλεσμα με τον αλγόριθμο προτεραιοτήτων να έχουν πολύ μικρές καθυστερήσεις, που σε ορισμένες περιπτώσεις είναι λίγο μικρότερες ακόμη και από το SJF. Όμως γενικά μπορούμε να πούμε ότι ο αλγόριθμος έχει ταυτόσημη συμπεριφορά με τον SJF για τους κόμβους 2 και 3. Το μόνο, ίσως, σημείο της γραφικής που ίσως φανεί παράξενο είναι η αύξηση της καθυστέρησης για $C=[1.0\ 5.5\ 1.5]$ αν και ο συνολικός ρυθμός εξυπηρέτησης του συστήματος αυξάνεται από το προηγούμενο πείραμα. Στην πραγματικότητα, όμως, ο ρυθμός εξυπηρέτησης που βλέπει ο κόμβος 2 στις ουρές των άλλων δύο μειώνεται αφού από $1+2.5=3.5$ γίνεται $1+1.5=2.5$. Έτσι, όπως είναι φυσιολογικό η καθυστέρηση θα ανέβει και θα γίνει περίπου ίση με αυτή του πειράματος με $C=[1.0\ 4\ 1.5]$ όπου πάλι ο 2 βλέπει τον ίδιο συνολικό ρυθμό εξυπηρέτησης.

Τέλος, κύριο χαρακτηριστικό του παρακάτω γραφήματος είναι συνεχώς μειούμενη διαφορά μεταξύ τω ν αλγορίθμων όσο αυξάνεται ο συνολικό ρυθμός εξυπηρέτησης του συστήματος, κάτι που είναι αναμενόμενο αφού το σύστημα είναι όλο και λιγότερο φορτωμένο και επομένως όλοι οι αλγόριθμοι καταφέρνουν να εξυπηρετήσουν ικανοποιητικά τις αιτήσεις των κόμβων.

Στην επόμενη σελίδα, φαίνεται το διάγραμμα καθυστερήσεων για κάθε κόμβο ξεχωριστά.



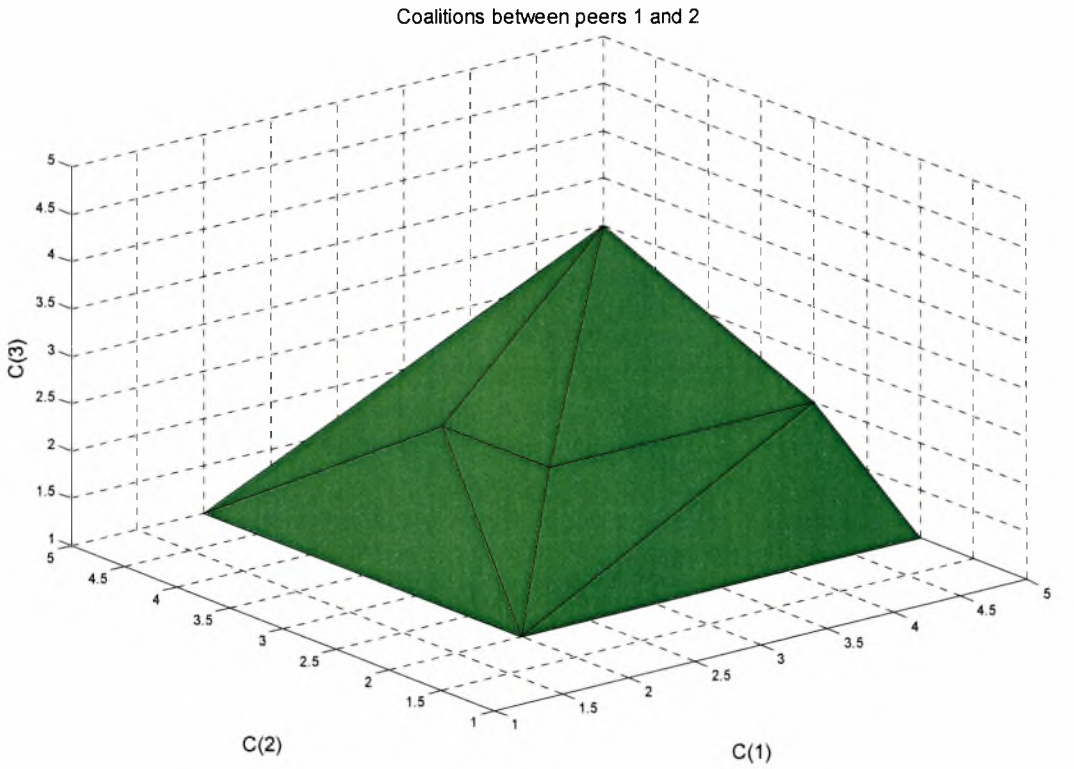
Όμως για να έχει κανείς μια καλή εικόνα για την απόδοση ενός αλγορίθμου είναι απαραίτητη η γνώση της συμπεριφοράς του σε όλο το σύστημα συνολικά. Για το λόγο αυτό παραθέτουμε στη συνέχεια και το διάγραμμα της συνολικής μέσης καθυστέρησης στο σύστημα.



Όπως παρατηρούμε στο γράφημα, ο FCFS δίνει τη μεγαλύτερη συνολική καθυστέρηση στο σύστημα. Ο αλγόριθμος δυναμικών προτεραιοτήτων έχει λίγο καλύτερα αποτελέσματα από τον FCFS. Στην περίπτωση αυτή, παρατηρούμε ότι αν και για τους κόμβους που σχηματίζουν τη συμμαχία η απόδοση είναι ταυτόσημη με αυτή του SJF, η συνολική εικόνα του δεν είναι η ίδια. Εξαιτίας της μεγάλης αύξησης στην καθυστέρηση του κόμβου 1, που δε συμμετέχει στη συμμαχία, ο μέσος όρος του συστήματος ανεβαίνει αρκετά πάνω από τον αλγόριθμο SJF, ο οποίος είναι όπως αναμενόταν ο καλύτερος. Τέλος, επιβεβαιώνουμε τη συνεχώς φθίνουσα πορεία του μέσου χρόνου ανάκτησης εξαιτίας του συνεχώς αυξανόμενου συνολικού ρυθμού εξυπηρέτησης του συστήματος αλλά και το γεγονός ότι οι αλγόριθμοι συγκλίνουν σταδιακά, γεγονός που οφείλεται, όπως είπαμε και προηγουμένως, στο συνεχώς μειούμενο φόρτο του συστήματος και στη μικρή διαφορά απόδοσης των αλγορίθμων στις περιπτώσεις αυτές.

III. Συμμαχία κόμβων (Coalition)

Ένα από τα κύρια χαρακτηριστικά του αλγορίθμου είναι η πιθανότητα εμφάνισης *συμμαχιών (coalitions)* μεταξύ των κόμβων με το μεγαλύτερο ρυθμό εξυπηρέτησης. Θυμίζουμε ότι με τον όρο *Συμμαχία (Coalition)* εννοούμε το φαινόμενο κατά το οποίο δύο κόμβοι δίνουν αμοιβαία προτεραιότητα στην εξυπηρέτηση των αιτημάτων τους σε βάρος της εξυπηρέτησης του τρίτου κόμβου του δικτύου. Για να ανιχνεύσουμε τις περιπτώσεις όπου εμφανίζεται συμμαχία μεταξύ δύο κόμβων για ένα συγκεκριμένο διάνυσμα ρυθμών εξυπηρέτησης `C`, ελέγχουμε τον πίνακα `reputation`. Όταν η τιμή φήμης που έχει ο ένας για τον άλλο ξεπερνά ένα κατώφλι, έστω t_1, και αυτό συμβαίνει για κάποιο μεγάλο ποσοστό των εποχών, έστω t_2, τότε θεωρούμε ότι για το συγκεκριμένο διάνυσμα `C` αναπτύχθηκε μια συμμαχία. Στο επόμενο διάγραμμα φαίνεται η περιοχή όπου εμφανίζεται coalition μεταξύ των κόμβων 1 και 2 για τιμές των ρυθμών εξυπηρέτησης εντός του διαστήματος [1,5]. Σημειώνουμε ότι η προσομοίωση έγινε για ρυθμό άφιξης κάθε κόμβου ίσο με $\lambda = 0.9$ ενώ τα κατώφλια ορίστηκαν σε $t_1 = 0.6$ και $t_2 = 0.7$.



Το διάγραμμα αυτό μας δίνει πολλά στοιχεία για τις περιπτώσεις όπου εμφανίζονται coalition μεταξύ δύο κόμβων. Αρχικά, παρατηρούμε ότι κύριο χαρακτηριστικό του χώρου αυτού είναι ότι οι τιμές των ρυθμών εξυπηρέτησης των κόμβων 1 και 2 είναι και οι δύο μεγαλύτερες από αυτόν του κόμβου 3. Έτσι, λοιπόν, βλέπουμε ότι, εξαιτίας του τρόπου που αποφασίζονται οι προτεραιότητας και τα σπασίματα στον αλγόριθμο, οι κόμβοι 1 και 2 αντιλαμβάνονται ότι ο κόμβος 3 είναι πολύ αργότερος από αυτούς και έτσι δεν έχουν κανένα κέρδος αν δώσουν σε αυτόν προτεραιότητα. Αντίθετα, ο κόμβος 2 βλέπει ότι αν δώσει προτεραιότητα στον κόμβο 1, θα ωθήσει και τον 1 στο να δίνει και εκείνος προτεραιότητα στα δικά του αιτήματα με αποτέλεσμα να μειώνεται κατά πολύ η μέση καθυστέρησή του. Όσο όμως

αυξάνεται ο ρυθμός εξυπηρέτησης του κόμβου 3, τα σημεία όπου έχουμε συμμαχία μεταξύ των δύο πρώτων κόμβων μειώνονται και μετά από την τιμή $C(3)=3$ παύουν να δημιουργούνται. Αυτό συμβαίνει γιατί πλέον ο κόμβος 3 έχει αρκετά μεγάλο ρυθμό εξυπηρέτησης ώστε να μη συμφέρει τους κόμβους 1 και 2 να συνεργαστούν. Αυτό επιβεβαιώνεται και από το γεγονός ότι η τιμή φήμης μεταξύ 1 και 2 πέφτει κάτω από το κατώφλι 0.6 το οποίο είχαμε ορίσει.

Τέλος, όπως είναι φυσιολογικό αντίστοιχα διαγράμματα προκύπτουν και για τις περιπτώσεις που δημιουργούνται coalition μεταξύ των 1 και 3 ή μεταξύ των κόμβων 2 και 3. Όμως, όπως είναι φυσιολογικό ο χώρος που εμφανίζονται αυτές θα βρίσκεται σε άλλη θέση του τρισδιάστατου χώρου που δημιουργείται για διάφορες τιμές των $C(1), C(2)$ και $C(3)$.

Σύγκριση Ευρεστικών και Βέλτιστων αλγορίθμων - Επίδραση εγωιστικής συμπεριφοράς

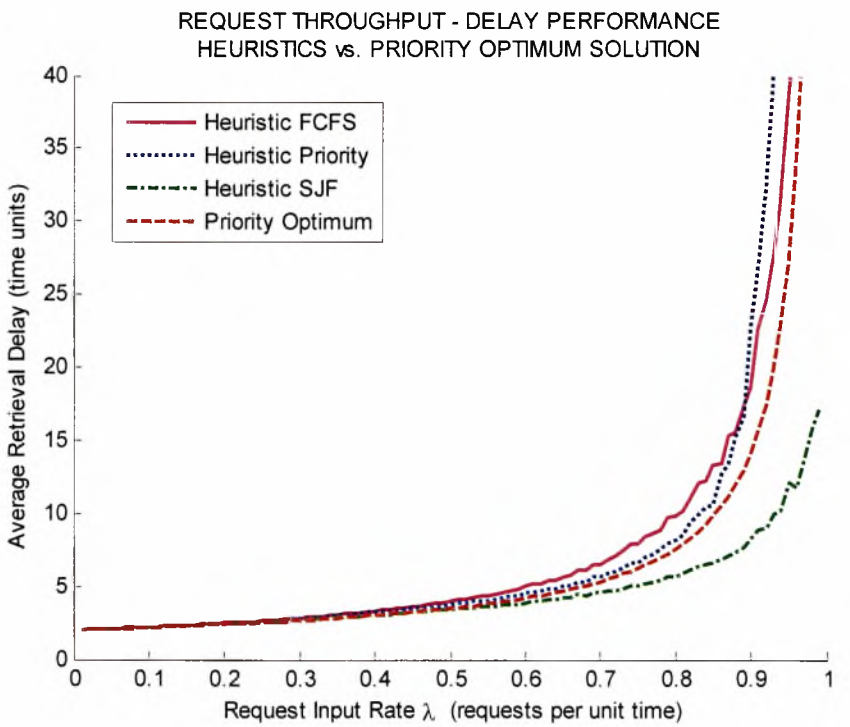
Στη συνέχεια θα αναλύσουμε και θα παρουσιάσουμε την επίδραση της εγωιστικής συμπεριφοράς που χαρακτηρίζει τους ευρεστικούς αλγορίθμους, τόσο κατά την επιλογή των σπασιμάτων τους, όσο και κατά την ανάθεση των προτεραιοτήτων για την περίπτωση του αλγορίθμου προτεραιοτήτων.

Για το σκοπό αυτό συγκρίνουμε την απόδοση των αλγορίθμων αυτών με τη βέλτιστη λύση που παρουσιάσαμε σε προηγούμενο εδάφιο. Ωστόσο, για να είναι δυνατή η σύγκριση θα χρησιμοποιήσουμε για όλες τις περιπτώσεις την βεβαρημένη μέση καθυστέρηση του συστήματος. Συγκεκριμένα η έκφραση που παρουσιάζουμε είναι η ακόλουθη

$$\sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} D_i = \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} \sum_{j=1, j\neq i}^{N} \frac{\lambda_{ij}}{\lambda_i} D_{ij} \qquad (4.9)$$

Η σύγκριση αυτή θα ήταν άδικη αν διατηρούσαμε τον ευρεστικό αλγόριθμο προτεραιοτήτων ως έχει, καθώς αντιμετωπίζει όλους τους χρήστες ως ίδιας σημασίας για το σύστημα. Για να αντιμετωπίσουμε αυτό το φαινόμενο τροποποιήσαμε το κριτήριο με το οποίο ο κάθε εξυπηρετητής ορίζει τις προτεραιότητες. Ορίσαμε λοιπόν ο κάθε εξυπηρετητής να ζυγίζει την καθυστέρηση που αντιμετωπίζει σε κάθε κόμβο j αντιστρόφως ανάλογα με το βάρος του b_j. Με αυτό τον τρόπο, κάθε κόμβος τείνει να δίνει πιο εύκολα προτεραιότητα στους κόμβους μεγαλύτερης βαρύτητας και έτσι επιτυγχάνουμε να ενσωματώσουμε και την έννοια των βαρών στον αλγόριθμων.

Στη συνέχεια παραθέτουμε το γράφημα που δείχνει την απόδοση των αλγορίθμων για διάφορες τιμές του ρυθμού παραγωγής αιτήσεων λ_i, ο οποίος είναι κοινός για όλους τους κόμβους. Στη συγκεκριμένη προσομοίωση όλοι οι κόμβοι έχουν ρυθμό εξυπηρέτησης $\mu=1$ και το διάνυσμα βαρών είναι το **b**=[3 2 1].



Όπως παρατηρούμε από το παραπάνω γράφημα για μικρές τιμές λ_i όλοι οι αλγόριθμοι έχουν παρόμοια απόδοση, ενώ καθώς το λ_i αυξάνει, αυξάνεται και η διαφορά στην απόδοση τους. Όπως ήταν αναμενόμενο η SJF πολιτική είναι η καλύτερη αφού εκμεταλλεύεται την επιπλέον γνώση του χρόνου εξυπηρέτησης κάθε εργασίας. Επίσης παρατηρούμε ότι πάντοτε η βέλτιστη λύση είναι καλύτερη των άλλων δύο και ειδικά για μεγάλες τιμές φόρτου η διαφορά είναι σημαντική. Ενδεικτικά παραθέτουμε την καθυστέρηση για τους δύο αλγορίθμους προτεραιοτήτων στον παρακάτω πίνακα.

λ_i / Αλγόριθμος	**0.6**	**0.7**	**0.8**	**0.9**
Ευρεστικός	4.565	5.705	8.195	22.76
Βέλτιστος	4.221	5.365	7.599	14.16

Η στρατηγική προτεραιοτήτων έχει ανάλογη απόδοση με την FCFS υπερτερώντας για μικρό φόρτο και υστερώντας όταν ο φόρτος στο σύστημα είναι ιδιαίτερα μεγάλος. Αυτό συμβαίνει γιατί οι δύο κόμβοι που δίνουν προτεραιότητα ο ένας στον άλλο εξασφαλίζουν μικρή καθυστέρηση για τους εαυτούς τους όμως ταυτόχρονα εκτινάσσουν την καθυστέρηση του τρίτου η οποία συμπαρασύρει και την συνολική καθυστέρηση του συστήματος.

Το συμπέρασμα λοιπόν στο οποίο καταλήγουμε είναι ότι η εγωιστική συμπεριφορά των κόμβων έχει ιδιαίτερα αρνητική επίδραση στη μέση καθυστέρηση του συστήματος, γεγονός που ενισχύεται καθώς αυξάνεται ο φόρτος στο σύστημα και καθώς εγκαθιδρύονται συνέργιες ανάμεσα σε κάποιους κόμβους.

Κεφάλαιο 5 – Πρόβλημα ελαχιστοποίησης μέγιστου χρόνου ανάκτησης

Το πρόβλημα ελαχιστοποίησης του μέγιστου χρόνου ανάκτησης βρίσκει εφαρμογή σε περιπτώσεις όπου ένας χρήστης στέλνει αιτήματα που αντιστοιχούν σε τμήματα κάποιας ολότητας, είτε αυτή είναι ένα αρχείο είτε κάποια άλλη εργασία που μπορεί να χωριστεί σε τμήματα. Όπως είναι κατανοητό, σε αυτή την περίπτωση δεν μας ενδιαφέρει ο μέσος χρόνος ανάκτησης αλλά ο μέγιστος από τους χρόνους εξυπηρέτησης αυτών των επιμέρους αιτημάτων. Έτσι λοιπόν, σε αντιστοιχία με όσα αναφέραμε για την επίλυση του προηγούμενου προβλήματος, το νέο πρόβλημα ελαχιστοποίησης αναφέρεται στην εύρεση ενός πίνακα «σπασιμάτων» Λ καθώς και ενός πίνακα προτεραιοτήτων Π για κάθε κόμβο ώστε να ελαχιστοποιείται το παρακάτω πρόβλημα:

$$\min_{\Lambda,\Pi} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij} \qquad (5.1)$$

$$\text{s.t.} \quad \sum_{j=1, j \neq i}^{N} \lambda_{ij} = \lambda_i \qquad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} > 0 \qquad \forall i, j$$

Η αντικειμενική συνάρτηση του παραπάνω προβλήματος είναι κυρτή (convex), αφού αποτελεί το σημείο προς σημείο μέγιστο κυρτών συναρτήσεων. Όμως, σε αντίθεση με το προηγούμενο πρόβλημα, δεν πρόκειται για συνεχή αντικειμενική συνάρτηση. Αυτό την καθιστά μη παραγωγίσιμη γεγονός που καθιστά δύσκολη την επίλυσή του με τις μεθόδους που χρησιμοποιήσαμε προηγουμένως. Ακόμη, δεν έχει αποδειχθεί ακόμη ότι και για το συγκεκριμένο πρόβλημα ισχύει ο κανόνας μC με αποτέλεσμα να μην μπορούμε να πούμε με σιγουριά ότι οι προτεραιότητες κάθε κόμβου καθορίζονται εξολοκλήρου από τα βάρη b_i ούτε ότι κάθε κόμβος έχει την ίδια προτεραιότητα σε κάθε εξυπηρετητή. Αυτό έχει σαν αποτέλεσμα το πρόβλημα (5.1) να μην μπορεί να αναχθεί σε ένα απλούστερο πρόβλημα εύρεσης μόνο του πίνακα Λ όπως είχε συμβεί στην προηγούμενη περίπτωση.

Παρά το γεγονός ότι δεν γνωρίζουμε αν ισχύει ο κανόνας μC εμείς θα υποθέσουμε ότι οι προτεραιότητες είναι σταθερές και προκαθορισμένες, και έστω σύμφωνα με τα βάρη b_i. Επομένως, θεωρούμε ότι ο πίνακας προτεραιοτήτων Π είναι γνωστός και ο μόνος άγνωστος στο πρόβλημα ελαχιστοποίησης είναι ο πίνακας των «σπασιμάτων» Λ.

Το πρόβλημα αυτό για να λυθεί πρέπει να μετατραπεί στην κανονική μορφή επίλυσης ενός κυρτού προβλήματος ελαχιστοποίησης. Αυτό μπορεί να γίνει εισάγοντας νέες μεταβλητές, έστω τις t_i, ώστε να αντικαταστήσουν τη συνάρτηση μεγίστου. Θέτοντας, λοιπόν, $t_i = \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} D_{ij}$ και προσθέτοντας ένα ακόμη περιορισμό, το πρόβλημα παίρνει τη μορφή:

$$\min_{\lambda_{ij}} \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} t_i \qquad (5.2)$$

$$s.t. \quad \sum_j \lambda_{ij} = \lambda_i$$

$$\frac{\lambda_{ij}}{\lambda_i} D_{ij} - t_i \le 0 \qquad \forall i, j$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \mathrm{j}$$

$$\lambda_{ij} \ge 0 \qquad \forall i, j$$

με $D_{ij} = \dfrac{1}{\left(1 - \sum_{k=1}^{i} \lambda_{kj}\right)\left(1 - \sum_{k=1}^{i-1} \lambda_{kj}\right)}$.

Ένας τρόπος επίλυσης του προβλήματος αυτού είναι με τη χρήση των συνθηκών ΚΚΤ. Για να γίνει αυτό πρέπει αρχικά να υπολογίσουμε τη συνάρτηση Lagrange, η οποία θα είναι η:

$$L = \sum_{i=1}^{N} b_i \frac{\lambda_i}{\lambda} t_i + \sum_{i=1}^{N} \xi_i \left(\lambda_i - \sum_{j=1, j\neq i}^{N} \lambda_{ij} \right) + \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \nu_{ij} \left(\frac{\lambda_{ij}}{\lambda_i \left(1 - \sum_{k=1}^{i} \lambda_{kj}\right)\left(1 - \sum_{k=1}^{i-1} \lambda_{kj}\right)} - t_i \right)$$

$$+ \sum_{j=1}^{N} \phi_j \left(1 - \sum_{j=1}^{N} \lambda_{ij} \right) \qquad (5.3)$$

Στη σχέση (5.3) παρατηρούμε ότι οι άγνωστοι είναι τα λ_{ij}, τα t_i και οι ΚΚΤ μεταβλητές ξ_i, ν_{ij} και ϕ_j.

Έπειτα, παραγωγίζουμε τη συνάρτηση Lagrange ως προς όλους αυτούς τους αγνώστους και θέτοντας κάθε μια μερική παράγωγο ίση με το μηδέν παίρνουμε τις εξισώσεις του συστήματος που πρέπει να λύσουμε για να βρούμε τη βέλτιστη λύση.

Εγωιστική Προσέγγιση

Περιγραφή Μεθόδου

Σε αντιστοιχία με την εγωιστική μέθοδο επίλυσης του προβλήματος ελαχιστοποίησης του μέσου χρόνου ανάκτησης και αντικαθιστώντας το D_{ij}, κάθε κόμβος i αποφασίζει τα λ_{ij} του έτσι ώστε να ελαχιστοποιήσει το παρακάτω πρόβλημα:

$$\min_{\lambda_{ij}} b_i \frac{\lambda_i}{\lambda} \max_{j \neq i} \frac{\lambda_{ij}}{\lambda_i} \frac{1}{\left(1-\sum_{k \leq i} \lambda_{kj}\right)\left(1-\sum_{k<i} \lambda_{kj}\right)} \tag{5.4}$$

$$\text{s.t.} \quad \sum_{j=1, j \neq i}^{N} \lambda_{ij} = \lambda_i \qquad \forall \text{i}$$

$$\sum_{i=1}^{N} \lambda_{ij} < 1 \qquad \forall \text{j}$$

$$\lambda_{ij} > 0 \qquad \forall i, j$$

Η αντικειμενική συνάρτηση του προβλήματος (5.4) είναι κυρτή (convex) αφού το σημείο προς σημείο μέγιστο κυρτών συναρτήσεων είναι κυρτό. Όμως η συνάρτηση μεγίστου δεν είναι συνεχής, και επομένως ούτε παραγωγίσιμη, με αποτέλεσμα να μην είναι δυνατή η επίλυσή του με καμία μέθοδο που χρησιμοποιεί μερικές παραγώγους. Το πρόβλημα αυτό δεν μπορεί να λυθεί ούτε με την κλασική μέθοδο Waterfilling που είχαμε περιγράψει στο Κεφάλαιο 4 αφού και αυτή κάνει χρήση των μερικών παραγώγων.

Παρόλα αυτά μπορούμε να βρούμε τη βέλτιστη λύση του εγωιστικού προβλήματος εφαρμόζοντας μια παραλλαγή της μεθόδου Waterfilling. Η παραλλαγή αυτή δεν χρησιμοποιεί καθόλου τις μερικές παραγώγους της αντικειμενικής συνάρτησης αλλά στηρίζεται σε ένα χαρακτηριστικό της βέλτιστης λύσης για κάθε κόμβο. Στη βέλτιστη λύση, οι καθυστερήσεις που αντιμετωπίζει κάθε χρήστης σε κάθε εξυπηρετητή θα είναι ίσες. Αν δε συνέβαινε αυτό, ο κόμβος θα προσπαθούσε να ελαττώσει λίγο το ρυθμό με τον οποίο στέλνει αιτήσεις στον εξυπηρετητή όπου έχει τη μέγιστη καθυστέρηση, ώστε να τη μειώσει, και να τον προσθέσει σε κάποιον άλλο στον οποίο έχει μικρότερη καθυστέρηση. Έτσι, στη βέλτιστη λύση ο κάθε κόμβος δε μπορεί με κανένα τρόπο να μειώσει τη μέγιστη καθυστέρηση που αντιμετωπίζει, αφού όλες οι καθυστερήσεις θα έχουν εξισωθεί. Επομένως, η μέθοδος επίλυσης που χρησιμοποιήσαμε προσπαθεί να εξισώσει τις καθυστερήσεις κάνοντας ανακατανομές μικρών ποσοτήτων ρυθμού παραγωγής αιτήσεων ίσες με ε από τον εξυπηρετητή με τη μέγιστη καθυστέρηση σε αυτόν με την ελάχιστη, αντίστοιχα με την κλασσική μέθοδο Waterfilling που προσπαθούσε να εξισώσει τις μερικές παραγώγους.

Πρωτόκολλο 1

Όπως φαίνεται από την έκφραση (5.4) η μόνη πληροφορία που χρειάζεται ο κόμβος i για να υπολογίσει την καθυστέρηση σε κάθε κόμβο και να εκτελέσει τον αλγόριθμο είναι η ποσότητα $\sum_{k<i}\lambda_{kj}$ για κάθε εξυπηρετητή j, δηλαδή το άθροισμα των σπασιμάτων των κόμβων μεγαλύτερης προτεραιότητας σε κάθε κόμβο j. Επίσης, η καθυστέρηση του κόμβου i δεν εξαρτάται από τα σπασίματα των κόμβων μικρότερης προτεραιότητας.

Βάσει των παραπάνω παρατηρήσεων, το πρωτόκολλο μπορεί να έχει ως εξής: Αρχικά, όλοι οι κόμβοι ανακοινώνουν στο δίκτυο τα βάρη τους b_i και επομένως κάθε κόμβος γνωρίζει την προτεραιότητα του σε σχέση με τους άλλους. Στη συνέχεια, οι κόμβοι με σειρά μειούμενης προτεραιότητας (η ισοδύναμα κατά μειούμενα b_i) εκτελούν τον παραπάνω αλγόριθμο. Ο κόμβος i λοιπόν δέχεται ένα διάνυσμα $\mathbf{Sum}_i = \left(\sum_{k<i}\lambda_{kj} : j = 1,...,N\right)$ από τον κόμβο $i-1$ και λύνει το παραπάνω πρόβλημα βελτιστοποίησης, αποφασίζει δηλαδή το διάνυσμα των σπασιμάτων του $\boldsymbol{\lambda}_i = \left(\lambda_{ij} : j = 1,\ldots,N, j \neq i\right)$, έτσι ώστε να μην παραβιάζεται η συνθήκη ευστάθειας του συστήματος σε κανέναν εξυπηρετητή. Αξίζει να σημειώσουμε ότι στη συνθήκη ευστάθειας, στην περίπτωση αυτή, το άνω όριο του αθροίσματος δεν είναι το N αλλά το i αφού οι υπόλοιποι κόμβοι δεν έχουν επιλέξει ακόμη τα δικά τους σπασίματα. Ωστόσο, σε περιπτώσεις που $\sum_{i=1}^{N}\lambda_i \simeq \sum_{j=1}^{N}C_j$ παρουσιάζεται το φαινόμενο ο κόμβος i να αδυνατεί να βρει ένα διάνυσμα $\boldsymbol{\lambda}_i$ ώστε να διατηρήσει το σύστημα σε ευστάθεια, με τις δεδομένες επιλογές των κόμβων μεγαλύτερης προτεραιότητας. Το φαινόμενο αυτό θα παρουσιαστεί αναλυτικότερα και στη συνέχεια. Τέλος, προσθέτει τα δικά του σπασίματα στο διάνυσμα Sum_i και έτσι δημιουργεί το Sum_{i+1} το οποίο και αποστέλλει στον $i+1$ κόμβο.

Αν οι κόμβοι εκτελέσουν τον αλγόριθμο με την προαναφερθείσα σειρά, απαιτούνται μόλις N εκτελέσεις του αλγορίθμου μία για κάθε κόμβο. Το αποτέλεσμα της παραπάνω διαδικασίας είναι ο εγωιστικός πίνακας σπασιμάτων Λ_ε και το $D_e = \left(D_{e,1},\ldots,D_{e,N}\right)$ είναι το βέλτιστο εφικτό διάνυσμα καθυστερήσεων του εγωιστικού αλγορίθμου.

Πρωτόκολλο 2

Ένα εναλλακτικό πρωτόκολλο θα μπορούσε να προϋποθέτει την πραγματοποίηση της ανανέωσης των σπασιμάτων κάθε κόμβου με τυχαία σειρά. Σε αυτή την περίπτωση ο κόμβος i εκτός της ποσότητας $\sum_{k<i}\lambda_{kj}$, χρειάζεται επιπλέον να γνωρίζει και την ποσότητα $\sum_{k=1}^{N}\lambda_{kj}$ για κάθε εξυπηρετητή j, ώστε να εξασφαλίζει την ευστάθεια του συστήματος. Επίσης, η καθυστέρηση του κόμβου i δεν εξαρτάται άμεσα από τα σπασίματα των κόμβων μικρότερης προτεραιότητας, παρά μόνο με την

έννοια ότι είναι πιθανό να περιορίζεται στις επιλογές του από την συνθήκη ευστάθειας του συστήματος.

Βάσει των παραπάνω παρατηρήσεων, το εναλλακτικό πρωτόκολλο μπορεί να έχει ως εξής: Το σύστημα ξεκινά από μία αρχική κατάσταση όπου κάθε κόμβος θεωρεί ένα τυχαίο πίνακα σπασιμάτων του συστήματος Λ^0. Σε μια τυχαία στιγμή ένας οποιοσδήποτε κόμβος i επιλέγει να εκτελέσει τον παραπάνω αλγόριθμο. Ο κόμβος i λοιπόν έχει τον πίνακα Λ στη διάθεση του και λύνει το παραπάνω πρόβλημα βελτιστοποίησης, αποφασίζει δηλαδή το διάνυσμα των σπασιμάτων του $\boldsymbol{\lambda}_i = \left(\lambda_{ij} : j = 1,\ldots,N, j \neq i\right)$, έτσι ώστε να μην παραβιάζεται η συνθήκη ευστάθειας του συστήματος σε κανέναν εξυπηρετητή. Ακολούθως, ανακοινώνει σε όλους τους άλλους κόμβους τα νέα σπασίματα που επέλεξε, δηλαδή το διάνυσμα Λ_i, οι οποίοι βάσει αυτού ενημερώνουν τον πίνακα Λ.

Το πρωτόκολλο που μόλις περιγράψαμε, όπως θα δούμε και στη συνέχεια αναλυτικότερα, μετά από έναν αριθμό εκτελέσεων του αλγορίθμου από κάθε κόμβο συγκλίνει, δηλαδή κανείς κόμβος πλέον δεν τροποποιεί τα σπασίματά του. Το αποτέλεσμα της παραπάνω διαδικασίας είναι ο εγωιστικός πίνακας σπασιμάτων Λ_ε και το $D_e = \left(D_{e,1},\ldots,D_{e,N}\right)$ είναι το βέλτιστο εφικτό διάνυσμα καθυστερήσεων του εγωιστικού αλγορίθμου για το πρωτόκολλο αυτό.

Προσομοίωση

Για την καλύτερη κατανόηση της συμπεριφοράς και την αξιολόγηση της απόδοσης των παραπάνω αλγορίθμων προχωρήσαμε στην υλοποίηση τους και στην προσομοίωση ενός δικτύου βάσει του μοντέλου που περιγράψαμε και προηγουμένως. Το σύνολο των προσομοιώσεων πραγματοποιήθηκε σε περιβάλλον MATLAB και γενικά, εκτός αν αναφέρεται διαφορετικά, το δίκτυο που μοντελοποιούμε αποτελείται από N=4 κόμβους με διάνυσμα βαρών το **b**=[4 3 2 1]. Επίσης, όλοι οι κόμβοι έχουν τον ίδιο ρυθμό παραγωγής αιτήσεων λ_i και το βήμα της τεχνικής waterfilling έχει τιμή $\varepsilon=10^{-3}$. Στη συνέχεια παρουσιάζουμε κάποια από τα αποτελέσματα που προέκυψαν.

Πρωτόκολλο 1

Απόδοση αλγορίθμου

Όπως έχουμε ήδη επισημάνει στο ορισμό του πρωτοκόλλου χρειάζεται μόλις μία εκτέλεση του αλγορίθμου για την εύρεση του βέλτιστου πίνακα σπασιμάτων. Ωστόσο, για μικρό αριθμό κόμβων και για μεγάλο φόρτο αιτήσεων στο σύστημα, παρατηρείται, όπως και στο προηγούμενο πρόβλημα, το φαινόμενο ο τελευταίος από τους κόμβους χαμηλής προτεραιότητας να αδυνατεί να βρει ένα διάνυσμα $\boldsymbol{\lambda}_i$ που να ικανοποιεί την συνθήκη ευστάθειας του συστήματος. Ενδεικτικά, θα αναφέρουμε ένα παράδειγμα όπου εμφανίζεται το παραπάνω φαινόμενο.

Αν έχουμε $\lambda_i = 0.95$ για κάθε κόμβο ο αλγόριθμος δίνει τον παρακάτω πίνακα σπασιμάτων.

Client \ Server	1	2	3	4
1	0	0.3167	0.3167	0.3167
2	0.4544	0	0.2479	0.2478
3	0.3072	0.4219	0	0.2209
4	0.2383	0.2613	-1	0

Σε αυτή τη λύση αντιστοιχεί ο παρακάτω πίνακας καθυστερήσεων.

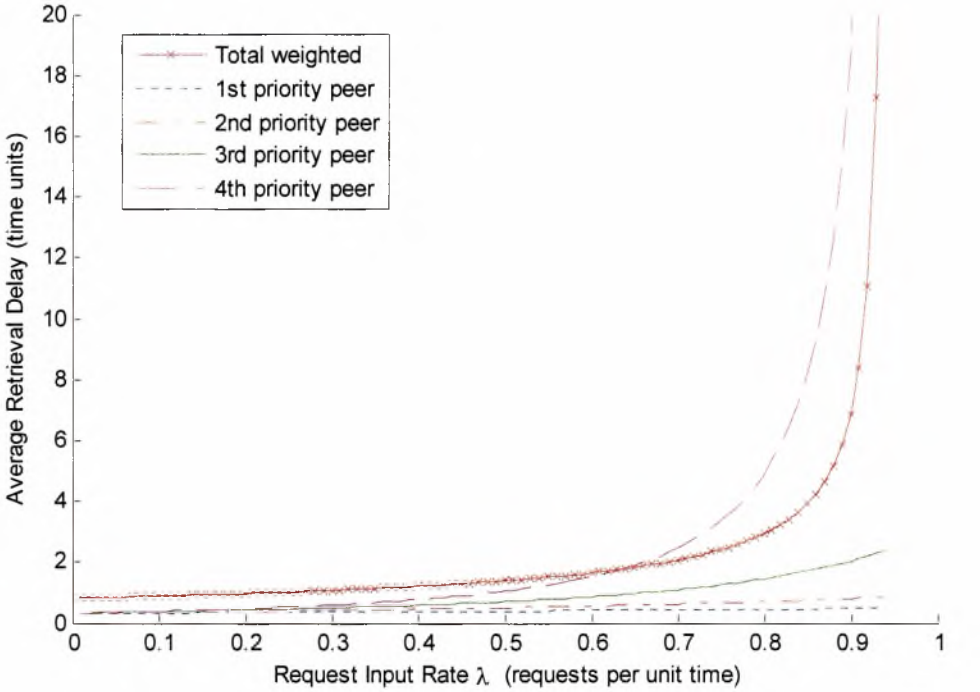
Client \ Server	1	2	3	4
1	0	0.4879	0.4877	0.4879
2	0.8766	0	0.8766	0.8763
3	2.4877	2.4861	0	2.4846
4	-1	-1	-1	0

Όπως, καταδεικνύει και ο παραπάνω πίνακας, αυτή την φορά δεν έχουμε ένα πρόβλημα εξισορρόπησης φόρτου (load balancing) όπως είχαμε προηγουμένως, αλλά ένα πρόβλημα εξισορρόπησης των καθυστερήσεων που βλέπει ο κάθε κόμβος στους υπολοίπους. Συγκεκριμένα βλέπουμε ότι κάθε κόμβος επιλέγει έτσι τα σπασίματα του ώστε να έχει την ίδια καθυστέρηση σε όλους τους κόμβους. Οι μικρές διακυμάνσεις στις καθυστερήσεις σε κάθε γραμμή οφείλονται στην ακρίβεια της μεθόδου waterfilling.

Ωστόσο, και πάλι το φαινόμενο εμφανίζεται λόγω της αδιαφορίας που επιδεικνύουν οι κόμβοι μεγαλύτερης προτεραιότητας για τους υπολοίπους, λόγω του ότι οι δεύτεροι δεν επηρεάζουν την καθυστέρηση τους. Έτσι λοιπόν στο παράδειγμά μας, οι τρεις πρώτοι κόμβοι κάνουν τις επιλογές τους με μοναδικό κριτήριο την καθυστέρηση που θα αντιμετωπίσουν και αγνοούν την ύπαρξη των κόμβων χαμηλότερης προτεραιότητας. Αυτό έχει ως αποτέλεσμα τελικά να παραμένει αδιάθετη χωρητικότητα στον εξυπηρετητή 4. Το γεγονός αυτό υποδηλώνεται στο παρακάτω γράφημα μέσω της έλλειψης σημείων για τα λ_i για τα οποία δεν βρέθηκε εφικτή λύση. Στο συγκεκριμένο παράδειγμα βλέπουμε ότι για τιμές $\lambda_i \geq 0.95$ ο αλγόριθμος δεν μπορεί να βρει λύση που να ικανοποιεί την συνθήκη ευστάθειας. Το φαινόμενο αυτό γενικά εμφανίζεται λιγότερες φορές καθώς αυξάνουμε τον αριθμό των κόμβων του συστήματος. Αυτό οφείλεται στο ότι επιτυγχάνεται καλύτερος διαμοιρασμός των αιτήσεων και παραμένει αδιάθετο μικρότερο ποσοστό χωρητικότητας στον τελευταίο εξυπηρετητή.

Στο γράφημα φαίνεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικός βεβαρημένος μέσος χρόνος ανάκτησης του συστήματος για διάφορες τιμές του ρυθμού παραγωγής αιτήσεων λ_i. Σημειώνουμε, ότι για να είναι πιο ευδιάκριτες οι γραφικές παραστάσεις, ιδιαίτερα σε μικρές τιμές των λ_i, στον κατακόρυφο άξονα εμφανίζεται μόνο το διάστημα [0-20] αν και οι γραφικές παραστάσεις της καθυστέρησης για τον τέταρτο κόμβο και του συνολικού βεβαρημένου αθροίσματος εκτείνονται και εκτός αυτού του διαστήματος.

OBJECTIVE 2: REQUEST THROUGHPUT - DELAY PERFORMANCE FOR EGOTISTIC APPROACH (P1)



Το πρώτο που παρατηρεί κανείς στο γράφημα είναι ότι καθώς αυξάνεται ο ρυθμός παραγωγής αιτήσεων, η καθυστέρηση αυξάνεται ιδιαίτερα. Επίσης είναι εμφανές ότι γενικά ο ρυθμός αύξησης της καθυστέρησης ενός κόμβου εξαρτάται από την προτεραιότητά του. Έτσι λοιπόν ο κόμβος 1 πού έχει απόλυτη προτεραιότητα δεν επηρεάζεται ιδιαίτερα από την αύξηση του λ_i, ενώ ο κόμβος 4 βλέπει την καθυστέρηση του να αυξάνει με ταχύτατο ρυθμό. Αιτία αυτού του φαινομένου είναι ότι ουσιαστικά ο κόμβος 1 έχοντας απόλυτη προτεραιότητα επηρεάζεται μόνο από την αύξηση του δικού του ρυθμού παραγωγής αιτήσεων, ενώ ο κόμβος 4 από την αύξηση όλων των λ_i. Γενικά κάθε κόμβος επηρεάζεται μόνο από την αύξηση των ρυθμών παραγωγής των κόμβων μεγαλύτερης προτεραιότητας.

Επίσης, η συνολική καθυστέρηση στο σύστημα, όντας ο βεβαρημένος μέσος όρος των καθυστερήσεων με βάρη μεγαλύτερα της μονάδας, θα περιμέναμε να βρίσκεται πάνω από τις επιμέρους καθυστερήσεις των κόμβων. Αυτό γενικά ισχύει, όχι όμως και για μεγάλες τιμές του λ_i. Σε αυτές τις περιπτώσεις η τιμή της συνολικής καθυστέρησης είναι μικρότερη από αυτή του κόμβου 4 λόγω της ραγδαίας αύξησης που περιγράψαμε παραπάνω.

Τέλος, επιβεβαιώνεται ότι οι κόμβοι μεγαλύτερης προτεραιότητας έχουν μικρότερη καθυστέρηση. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.9$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα

D_1	D_2	D_3	D_4	Total Delay
0.4763	0.8181	2.033	18.98	6.851

Πρωτόκολλο 2

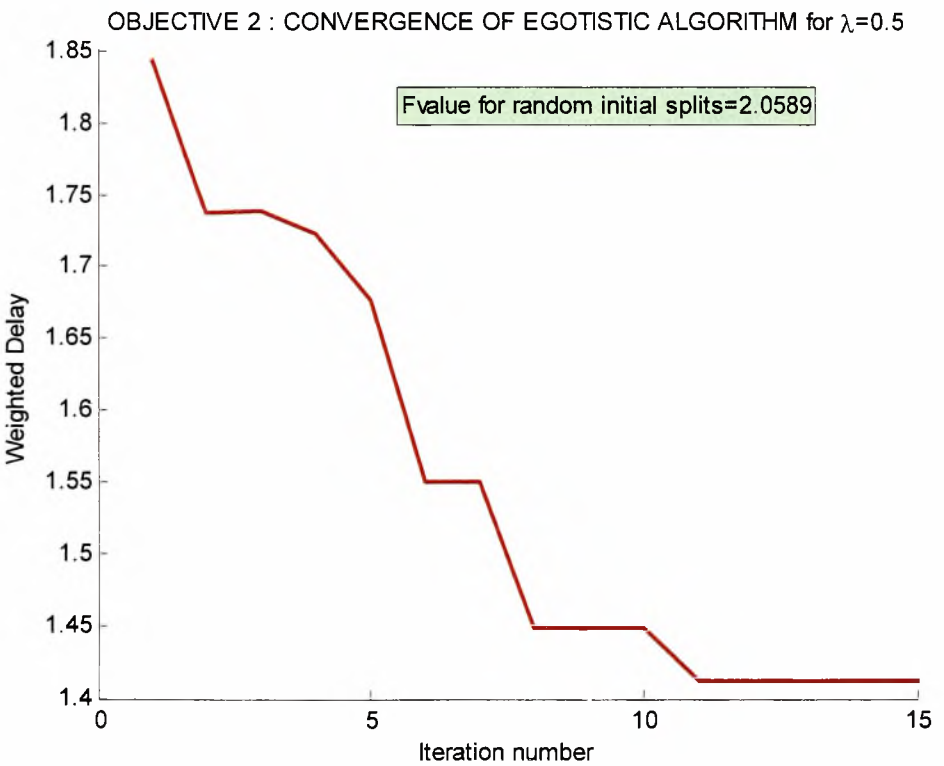
I. Σύγκλιση αλγορίθμου

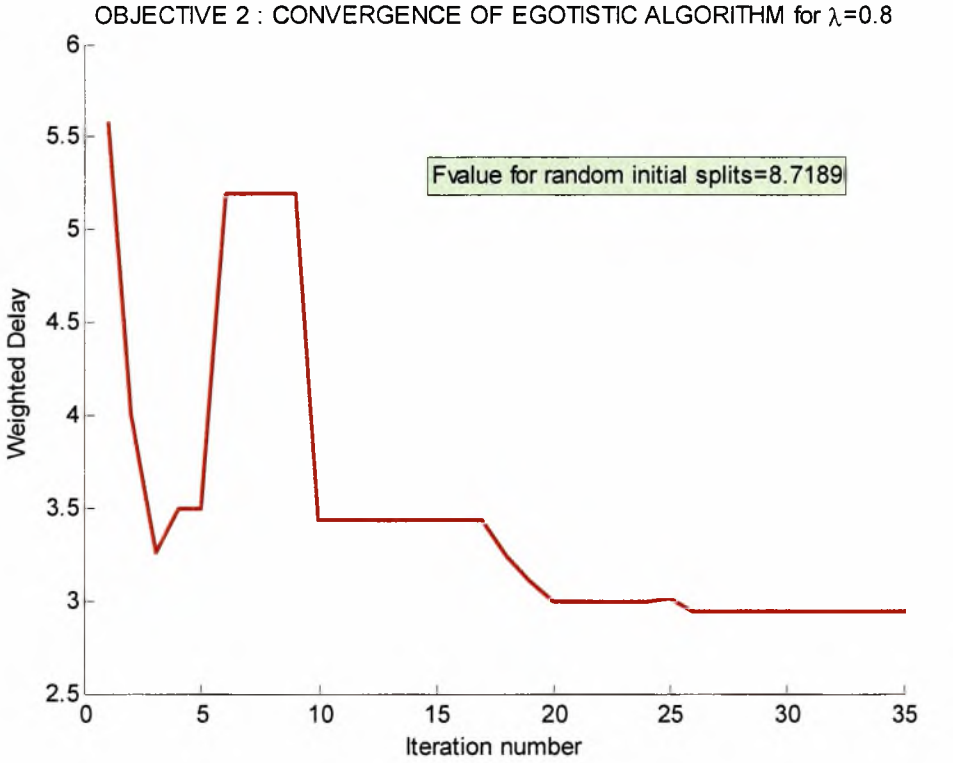
Το πρωτόκολλο αυτό μας δίνει την δυνατότητα η ανανέωση των σπασιμάτων κάθε κόμβου να πραγματοποιείται με τυχαία σειρά, ενώ ταυτόχρονα λόγω της ύπαρξης του αρχικού τυχαίου πίνακα σπασιμάτων Λ^0 και της συνθήκης ευστάθειας εξασφαλίζει ότι το σύστημα θα εντοπίσει μια εφικτή λύση σε κάθε περίπτωση όπου $\sum_{i=1}^{N}\lambda_i < \sum_{j=1}^{N}\mu_j$.

Η δυνατότητα αυτή που αναφέραμε παραπάνω δεν μας παρέχεται χωρίς κόστος. Το κόστος λοιπόν είναι ότι πλέον δεν αρκεί μια εκτέλεση του αλγορίθμου για κάθε κόμβο, αλλά η διαδικασία αυτή πρέπει να επαναλαμβάνεται. Ωστόσο, μετά από έναν αριθμό εκτελέσεων του αλγορίθμου από κάθε κόμβο επιτυγχάνεται σύγκλιση, δηλαδή κανείς κόμβος πλέον δεν τροποποιεί τα σπασίματά του. Είναι αυτονόητο ότι καθώς ο αριθμός των κόμβων του συστήματος αυξάνεται απαιτούνται περισσότερες επαναλήψεις ώστε να επιτευχθεί η σύγκλιση.

Στη συνέχεια, παρουσιάζουμε την ταχύτητα σύγκλισης του αλγορίθμου για δύο διαφορετικές περιπτώσεις ρυθμών αφίξεων λ για να δούμε πως συμπεριφέρεται ο αλγόριθμος για διαφορετικό φόρτο αιτήσεων.

Σε αυτό το σημείο είναι απαραίτητο να αναφέρουμε ότι εξαιτίας και της τυχαίας επιλογής του αρχικού πίνακα σπασιμάτων Λ^0, ο αριθμός των επαναλήψεων που απαιτούνται για την επίτευξη της σύγκλισης δεν είναι σταθερός. Ωστόσο, στα παρακάτω γραφήματα παρουσιάζουμε μια μέση περίπτωση όπου μπορεί κανείς να βγάλει ασφαλή συμπεράσματα για την ταχύτητα σύγκλισης του αλγορίθμου.



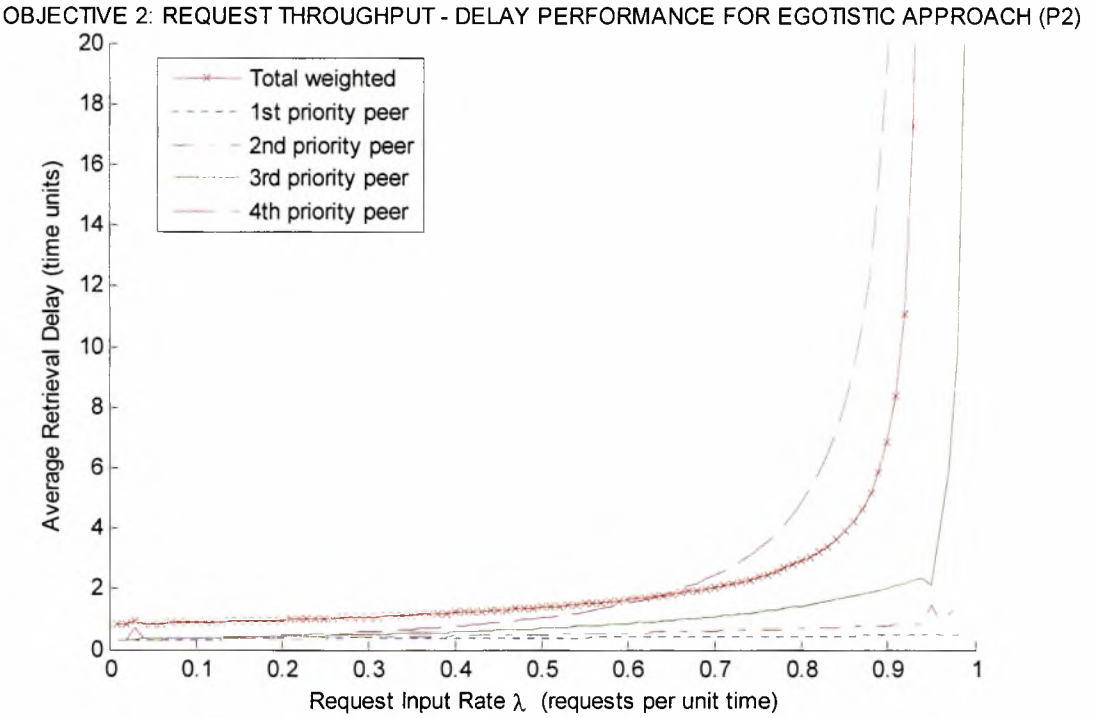


Όπως φαίνεται και από τα γραφήματα η μέση καθυστέρηση του συστήματος δεν μειώνεται πάντα μεταξύ δύο επαναλήψεων. Αυτό οφείλεται στο γεγονός ότι ο κάθε χρήστης προσπαθεί να ελαχιστοποιήσει την δικιά του καθυστέρηση αδιαφορώντας για την καθυστέρηση των άλλων. Παρ' όλα αυτά τελικά μετά από έναν αριθμό επαναλήψεων ο αλγόριθμος συγκλίνει και μάλιστα όπως είδαμε και προηγουμένως συγκλίνει στη λύση που δίνει και το πρωτόκολλο 1. Πιο συγκεκριμένα, οι κορυφές που εμφανίζονται στα γραφήματα οφείλονται στην εκτέλεση του αλγορίθμου από κάποιον κόμβο μεγάλης προτεραιότητας. Σε αυτές τις περιπτώσεις ο συγκεκριμένος κόμβος επιλέγει το καλύτερο διάνυσμα σπασιμάτων με κριτήριο τη δικιά του καθυστέρηση και μόνο. Αυτό, όμως μπορεί να έχει ως αποτέλεσμα να μην γίνεται καλός διαμοιρασμός του φόρτου στο σύστημα και σε κάποιον εξυπηρετητή να έχουμε λειτουργία στα όρια της περιοχής ευστάθειας. Συνεπώς η καθυστέρηση των κόμβων χαμηλής προτεραιότητας αυξάνεται πολύ συμπαρασύροντας και την συνολική καθυστέρηση στο σύστημα. Ενδεικτικό του γεγονότος αυτού είναι και το ότι στο δεύτερο γράφημα, όπου ο φόρτος είναι μεγαλύτερος παρουσιάζονται μεγαλύτερες διακυμάνσεις.

Λόγω των παραπάνω, στην πρώτη περίπτωση, όπου λ=0.5, ο αλγόριθμος χρειάζεται μόλις 12 επαναλήψεις για να συγκλίνει ενώ λ=0.8 η σύγκλιση επιτυγχάνεται γύρω στην τριακοστή επανάληψη. Επίσης, στο πλαίσιο εμφανίζεται η καθυστέρηση του συστήματος για τον τυχαίο αρχικό πίνακα σπασιμάτων Λ^0. Η τιμή αυτή υποδεικνύει την βελτίωση που επιτυγχάνεται με την χρήση του αλγορίθμου για την εύρεση των σπασιμάτων σε σχέση με μια τυχαία επιλογή σπασιμάτων.

II. Απόδοση αλγορίθμου

Όπως και για το προηγούμενο πρωτόκολλο λοιπόν ,στο παρακάτω γράφημα παρουσιάζεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικός βεβαρημένος μέσος χρόνος ανάκτησης του συστήματος για διάφορες τιμές του ρυθμού παραγωγής αιτήσεων λ_i. Και σε αυτή την περίπτωση πρέπει να αναφέρουμε, ότι για να είναι πιο ευδιάκριτες οι γραφικές παραστάσεις, ιδιαίτερα σε μικρές τιμές των λ_i, στον κατακόρυφο άξονα εμφανίζεται μόνο το διάστημα [0-20] αν και οι γραφικές παραστάσεις της καθυστέρησης για τον τέταρτο κόμβο και του συνολικού βεβαρημένου αθροίσματος εκτείνονται και εκτός αυτού του διαστήματος.



Πέραν των παρατηρήσεων που επισημάναμε στο πρωτόκολλο 1, οι οποίες ισχύουν και στην περίπτωση αυτού του πρωτοκόλλου, παρατηρούμε ότι πλέον είναι δυνατή η εύρεση λύσης και για τις περιπτώσεις όπου $\lambda_i \geq 0.95$. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.9$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα.

D_1	D_2	D_3	D_4	**Total Delay**
0.4764	0.8181	2.033	19.000	6.856

Συγκρίνοντας με τον πίνακα του πρωτοκόλλου 1 διαπιστώνουμε ότι η απόδοση των δύο πρωτοκόλλων είναι σχεδόν ταυτόσημη για τις τιμές των λ_i που βρίσκουν εφικτή λύση. Αυτό ισχύει γενικά, ανεξάρτητα από την σειρά με την οποία κόμβοι εκτελούν τις ανανεώσεις των σπασιμάτων τους (έχουμε ήδη αναφέρει ότι η σειρά είναι τυχαία) και επίσης ανεξάρτητα από τον αρχικό πίνακα σπασιμάτων Λ^0.

Η διαφορά έγκειται στο ότι το δεύτερο πρωτόκολλο βρίσκει μια εφικτή λύση και για τις άλλες περιπτώσεις έστω και αν αυτή χαρακτηρίζεται από πολύ μεγάλη καθυστέρηση.

Ειδικότερα για $\lambda_i = 0.95$ ο αλγόριθμος επιστρέφει τώρα τον παρακάτω πίνακα σπασιμάτων

Client \ Server	1	2	3	4
1	0	0.3049	0.3229	0.3222
2	0.4576	0	0.2456	0.2468
3	0.3324	0.3790	0	0.2386
4	0.2070	0.3141	0.4290	0

Σε αυτή τη λύση αντιστοιχεί ο παρακάτω πίνακας καθυστερήσεων.

Client \ Server	1	2	3	4
1	0	0.4618	0.5019	0.5004
2	0.8882	0	0.8846	0.8893
3	3.0724	1.8161	0	3.0284
4	345.864	522.9853	404.1833	0

Όπως, καταδεικνύει και ο παραπάνω πίνακας η εξισορρόπηση των καθυστερήσεων δεν έχει γίνει με ιδιαίτερα καλό τρόπο, όμως, αντίθετα με προηγουμένως, κατάφερε να εξασφαλιστεί η ευστάθεια του συστήματος.

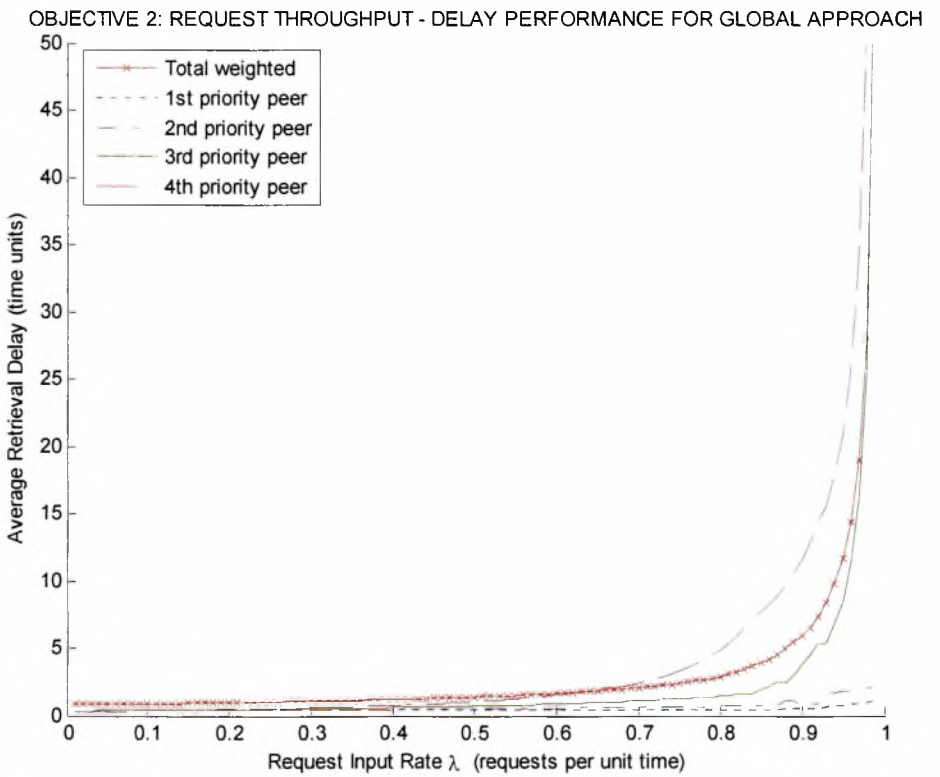
Καθολικό πρόβλημα

Για την καλύτερη κατανόηση της συμπεριφοράς και την αξιολόγηση της απόδοσης των μεθόδων επίλυσης του προβλήματος προχωρήσαμε στην προσομοίωση του καθολικού τρόπου επίλυσης χρησιμοποιώντας της συνθήκες ΚΚΤ. Η διαδικασία που ακολουθήθηκε ήταν ο υπολογισμός της συνάρτησης Lagrange και, στη συνέχεια, η χρήση των μερικών παραγώγων της, σε συνδυασμό με τις συνθήκες ΚΚΤ, ώστε να παραχθεί ένα σύστημα εξισώσεων από το οποίο θα προέκυπτε η βέλτιστη λύση. Η διαδικασία για τον υπολογισμό της συνάρτησης Lagrange (σχέση (5.3)) περιγράφεται στην αρχή του τρέχοντος κεφαλαίου. Το σύνολο των προσομοιώσεων πραγματοποιήθηκε σε περιβάλλον MATLAB και γενικά, εκτός αν αναφέρεται διαφορετικά, το δίκτυο που μοντελοποιούμε αποτελείται από N=4 κόμβους με διάνυσμα βαρών το **b**=[4 3 2 1]. Επίσης, όλοι οι κόμβοι έχουν τον ίδιο ρυθμό παραγωγής αιτήσεων λ_i.

Στη συνέχεια παρουσιάζουμε κάποια από τα αποτελέσματα που προέκυψαν.

Απόδοση αλγορίθμου

Στο παρακάτω γράφημα φαίνεται ο μέσος χρόνος ανάκτησης για κάθε κόμβο καθώς και ο συνολικό μέσος χρόνος ανάκτησης του συστήματος.



Ιδιαίτερο χαρακτηριστικό της γραφικής παράστασης είναι η μεγάλη αύξηση της καθυστέρησης καθώς αυξάνεται ο ρυθμός άφιξης αιτήσεων λ. Αυτό είναι απολύτως φυσιολογικό αφού ο φόρτος του συστήματος αυξάνεται καθώς μεγαλώνει το λ κάθε

κόμβου, προκαλώντας μεγαλύτερη καθυστέρηση, και ιδιαίτερα όταν πλησιάζουμε στο ρυθμό εξυπηρέτησης C, όπου το σύστημα αγγίζει τα όρια ευστάθειάς του. Ένα άλλο σημαντικό χαρακτηριστικό του συστήματος, το οποίο έχουμε την ευκαιρία να επιβεβαιώσουμε με αυτή τη γραφική, είναι η πολύ μεγάλη καθυστέρηση που αντιμετωπίζει ο κόμβος 4, εκείνος δηλαδή με τη μικρότερη προτεραιότητα. Αυτό συμβαίνει γιατί ο αλγόριθμος δίνει πολύ μεγάλη βαρύτητα στην ελαχιστοποίηση του χρόνου ανάκτησης των κόμβων με μεγάλη προτεραιότητα, ιδιαίτερα του κόμβου 1, γεγονός που λειτουργεί αρνητικά για την καθυστέρηση των κόμβων με μικρότερη προτεραιότητα, και ιδιαίτερα του κόμβου 4. Άλλωστε, είναι εύκολο να αντιληφθούμε ότι ο κόμβος 1 έχοντας απόλυτη προτεραιότητα επηρεάζεται μόνο από την αύξηση του δικού του ρυθμού παραγωγής αιτήσεων, ενώ ο κόμβος 4 από την αύξηση όλων των λ_i. Ως επιβεβαίωση των παραπάνω, παρατηρούμε ότι οι κόμβοι με μεγάλη προτεραιότητα, δηλαδή οι 1 και 2, έχουν σχεδόν μηδενική καθυστέρηση (ο κόμβος 1 έχει, όπως ήταν φυσιολογικό, μικρότερη καθυστέρηση από τον 2) ενώ οι κόμβοι 3 και 4 έχουν αρκετά μεγαλύτερη. Ενδεικτικά αναφέρουμε ότι για $\lambda_i = 0.95$ οι τιμές των καθυστερήσεων δίνονται στον παρακάτω πίνακα

D_1	D_2	D_3	D_4	**Total Delay**
0.7994	1.782	8.578	20.9	8.578

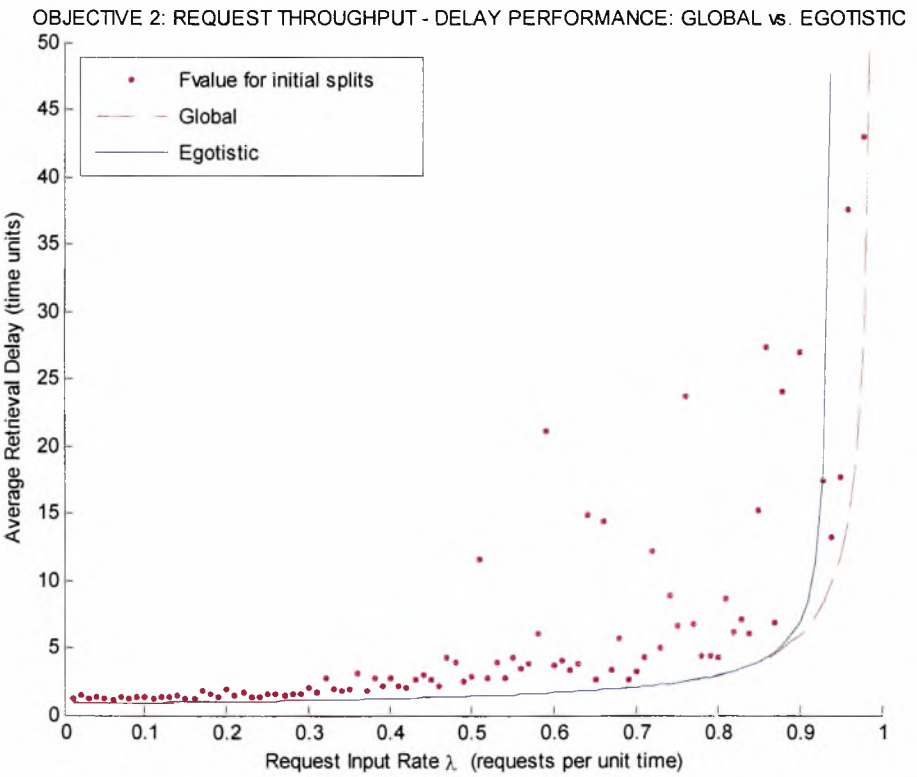
Τέλος, η συνολική καθυστέρηση στο σύστημα, όντας ο βεβαρημένος μέσος όρος των καθυστερήσεων με βάρη μεγαλύτερα της μονάδας, θα περιμέναμε να βρίσκεται πάνω από τις επιμέρους καθυστερήσεις των κόμβων. Αυτό γενικά ισχύει, όχι όμως και για μεγάλες τιμές του λ_i. Σε αυτές τις περιπτώσεις η τιμή της συνολικής καθυστέρησης είναι μικρότερη από αυτή του κόμβου 4 λόγω της ραγδαίας αύξησης που περιγράψαμε παραπάνω.

Σύγκριση απόδοσης αλγορίθμων

Όπως αναφέραμε και προηγουμένως, δεν ήταν εφικτό να προσομοιώσουμε την αλτρουιστική μέθοδο επίλυσης του προβλήματος λόγω των εγγενών δυσκολιών επίλυσης που παρουσιάζει. Παρόλα αυτά, ήταν δυνατή η προσομοίωση του αλγορίθμου επίλυσης του καθολικού προβλήματος, την οποία και θα χρησιμοποιήσουμε και για να συγκρίνουμε την εγωιστική μέθοδο επίλυσης με τη βέλτιστη λύση.

Το πρωτόκολλο 2 της εγωιστικής προσέγγισης καταφέρνει να βρει εφικτά διανύσματα για περιπτώσεις με πολύ μεγάλους ρυθμούς λ_i, όπου το πρωτόκολλο 1 αδυνατεί. Όμως στις περιπτώσεις αυτές οι λύσεις εμφανίζουν πολύ μεγάλη συνολική καθυστέρηση. Το φαινόμενο αυτό οφείλεται στο γεγονός ότι ο αλγόριθμος, αν και καταφέρνει να δώσει ένα εφικτό διάνυσμα λύσης, δεν μπορεί να κάνει αποτελεσματική ελαχιστοποίηση του συνολικού βεβαρημένου αθροίσματος των καθυστερήσεων. Για το λόγο αυτό, τα αποτελέσματα της σύγκρισης μεταξύ της λύσης του πρωτοκόλλου 2 και αυτής του καθολικού προβλήματος σε τέτοιες μεγάλες τιμές θα αλλοίωναν την πραγματική διαφορά των αλγορίθμων. Έτσι, στη σύγκριση του εγωιστικού αλγορίθμου με τον αλγόριθμο επίλυσης του καθολικού προβλήματος θα χρησιμοποιηθεί το πρωτόκολλο 1, αφού για τις περιπτώσεις που μπορεί να βρει κάποιο εφικτό διάνυσμα λύσης έχει απόδοση ίδια με αυτή του πρωτοκόλλου 2 (όπως αναφέραμε και σε προηγούμενο εδάφιο).

Στο παρακάτω γράφημα φαίνονται τα συνολικά βεβαρημένα αθροίσματα των δύο αλγορίθμων για την περίπτωση του δικτύου των τεσσάρων κόμβων και για ρυθμούς αφίξεων από 0.1 μέχρι 0.99 καθώς και οι καθυστερήσεις για την αρχική λύση από την οποία ξεκίνησαν οι αλγόριθμοι.



Όπως έχουμε εξηγήσει και σε προηγούμενο εδάφιο, ο εγωιστικός αλγόριθμος δεν καταφέρνει να βρει ένα εφικτό διάνυσμα για πολύ μεγάλους ρυθμούς αφίξεων και για αυτό το λόγο η γραφική του εγωιστικού αλγορίθμου δεν έχει τιμές για λ_i μεγαλύτερο από 0.94. Επίσης, δεν απεικονίζονται στο γράφημα τιμές καθυστέρησης μεγαλύτερες από 100, που αντιστοιχούν σε μεγάλα λ_i, για να είναι πιο ευδιάκριτες οι διαφορές των αλγορίθμων στις υπόλοιπες τιμές. Ενδεικτικά αναφέρουμε τις παρακάτω τιμές για κάποια από τα λ_i της προσομοίωσης.

λ_i / Αλγόριθμος	**0.5**	**0.7**	**0.8**	**0.9**	**0.94**
Εγωιστικός	1.415	2.09	2.957	6.915	47.65
Βέλτιστη λύση	1.408	2.082	2.939	5.921	9.79

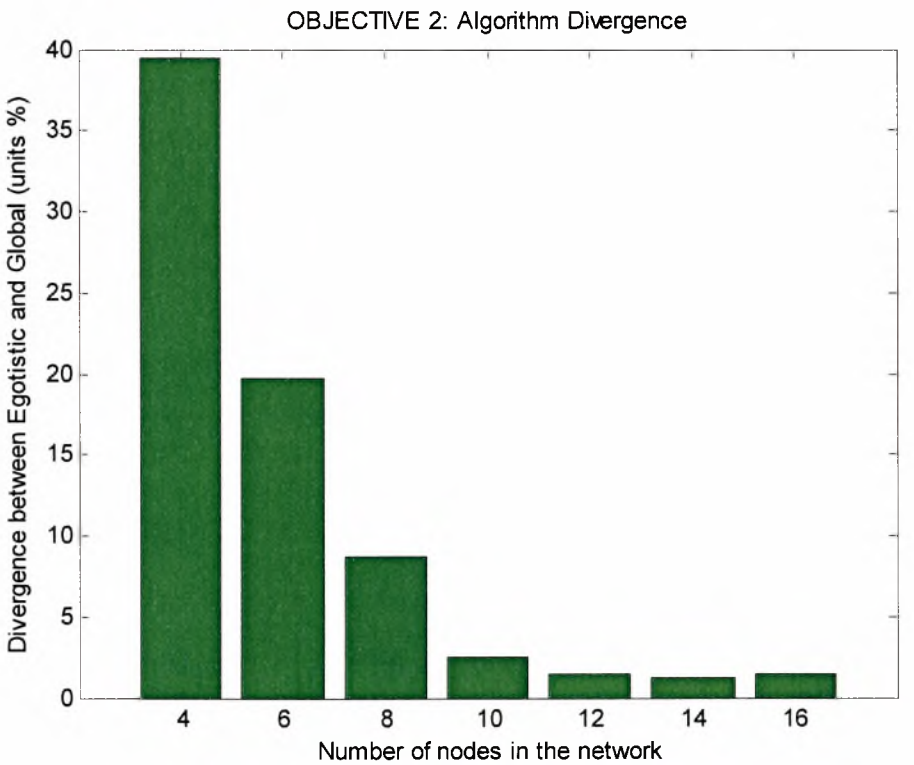
Γενικά, παρατηρούμε ότι για μικρές τιμές λ_i οι καθυστερήσεις που μας δίνουν οι αλγόριθμοι δεν απέχουν πολύ, ενώ για τιμές πάνω από το 0.85 περίπου ο εγωιστικός αλγόριθμος αρχίζει να μας δίνει σημαντικά χειρότερα αποτελέσματα. Αυτή η συμπεριφορά κορυφώνεται για $\lambda_i = 0.94$ όπου ο εγωιστικός αλγόριθμος καταφέρνει μετά βίας να βρει ένα εφικτό διάνυσμα, με πολύ μεγάλη όμως καθυστέρηση. Ο λόγος που συμβαίνει αυτό είναι ότι όσο το σύστημα λειτουργεί με μικρό φόρτο η λήψη αποφάσεων με βάση καθαρά εγωιστικά κριτήρια δεν επηρεάζει πολύ αρνητικά την καθυστέρηση, και μάλιστα η εγωιστική επιλογή των σπασιμάτων αποδεικνύεται μια σχεδόν βέλτιστη στρατηγική. Όταν όμως τα λ_i αρχίσουν να μεγαλώνουν, η εγωιστική στρατηγική δεν έχει το ίδιο καλά αποτελέσματα για το συνολικό δίκτυο. Στις περιπτώσεις αυτές οι επιλογές των κόμβων με μεγάλη προτεραιότητα περιορίζουν πολύ τις επιλογές των υπολοίπων, οι οποίοι αδυνατούν να σπάσουν τα αιτήματά τους με τον τρόπο θα ήθελαν και αναγκάζονται να καταφύγουν σε λύσεις που, αν και είναι οι καλύτερες δυνατές με τις συγκεκριμένες συνθήκες, αυξάνουν πολύ τη συνολική καθυστέρηση. Χαρακτηριστικό είναι ότι για πολύ μεγάλες τιμές λ_i (μεγαλύτερες από 0.90) η αρχική τυχαία τιμή μπορεί έχει συνολικά μικρότερη καθυστέρηση από ότι η τελική λύση του εγωιστικού αλγορίθμου. Όμως, η εγωιστική στρατηγική που ακολουθούν οι κόμβοι, με σημαντικότερη αυτή του πρώτου και δεύτερου κόμβου (λόγω της επιρροής τους στις καθυστερήσεις των άλλων) , οδηγεί σε χειρότερο αποτέλεσμα για το συνολικό δίκτυο.

Συγκρίνοντας τις καθυστερήσεις για τις τυχαίες αρχικές τιμές με τις τελικές τιμές των αλγορίθμων, βλέπουμε ότι για μικρά λ_i δεν έχουν μεγάλη διαφορά από τα αποτελέσματα των δύο αλγορίθμων, αν και πάντα είναι χειρότερα και από τους δύο. Θα μπορούσε, λοιπόν, κάποιος να αμφισβητήσει την αποτελεσματικότητα των αλγορίθμων. Όμως, ένας αλγόριθμος δείχνει την απόδοσή του για τιμές λ_i σχετικά μεγάλες. Σε τέτοιες τιμές λ_i, λοιπόν, βλέπουμε ότι το τελικό αποτέλεσμα του εγωιστικού αλγορίθμου είναι πάντα αρκετά καλύτερο από την αρχική τυχαία λύση, ενώ φυσικά η βέλτιστη λύση του καθολικού προβλήματος είναι ακόμη καλύτερη από αυτή της εγωιστικής προσέγγισης. Τέλος, για μεγάλα λ_i ο εγωιστικός μπορεί να είναι κάποιες φορές χειρότερος, φαινόμενο που δεν παρατηρείται σε καμία περίπτωση για το καθολικό πρόβλημα.

Στη συνέχεια, παραθέτουμε ένα γράφημα με τις ποσοστιαίες διαφορές των δύο αλγορίθμων για διαφορετικούς αριθμούς κόμβων στο δίκτυο. Για κάθε αριθμό κόμβων τρέξαμε την προσομοίωση για όλα τα λ_i εντός του διαστήματος [0.8, 0.99] εκτός από εκείνα όπου ο εγωιστικός αλγόριθμος αδυνατούσε να λύσει το πρόβλημα. Για κάθε διαφορετική τιμή λ_i, υπολογίζαμε το ποσοστό βελτίωσης που επιφέρει ο αλτρουιστικός αλγόριθμος επί του αποτελέσματος του εγωιστικού, δηλαδή το ποσοστό που δηλώνει ο τύπος:

$$ratio = \frac{egotistic - global}{global} \tag{5.5}$$

Στη συνέχεια, υπολογίζουμε το μέσο όρο όλων των τιμών λ_i για ένα συγκεκριμένο αριθμό κόμβων και τα παρουσιάζουμε με τη μορφή ράβδων.



Βλέποντας το παραπάνω γράφημα μπορούμε να πούμε γενικά ότι υπάρχει μια σαφής τάση μείωσης της διαφοράς των δύο αλγορίθμων. Αρχικά, για το δίκτυο των τεσσάρων κόμβων βλέπουμε ότι ο εγωιστικός αλγόριθμος δίνει περίπου 40% χειρότερη λύση από τη βέλτιστη λύση του καθολικού . Αυτό συμβαίνει εξαιτίας της κακής συμπεριφοράς του αλγορίθμου στα μεγάλα λ_i, όπως εξηγήσαμε και παραπάνω. Στη συνέχεια όμως, όσο αυξάνεται ο αριθμός των κόμβων ο αλγόριθμος καταφέρνει να λύσει καλύτερα το πρόβλημα ελαχιστοποίησης με αποτέλεσμα να συγκλίνει, για μεγάλο αριθμό κόμβων, στην απόδοση του εγωιστικού.

Ο λόγος που η εγωιστική προσέγγιση συγκλίνει τελικά στη λύση του καθολικού προβλήματος οφείλεται στους ίδιους λόγους που συνέβαινε το ίδιο στην περίπτωση του προβλήματος ελαχιστοποίησης του μέσου χρόνου ανάκτησης. Η μόνη διαφορά είναι ότι σε αυτή την περίπτωση πρόκειται για ένα πρόβλημα *Εξισορρόπησης Καθυστερήσεων* . Συγκεκριμένα παρατηρούμε ότι κάθε κόμβος επιλέγει έτσι τα

σπασίματα του ώστε να έχει την ίδια καθυστέρηση σε όλους τους κόμβους,. Όμως και σε αυτή την περίπτωση παραμένει αχρησιμοποίητο ένα μέρος του ρυθμού εξυπηρέτησης του τελευταίο κόμβου με αποτέλεσμα να υπάρχει αυτή η διαφορά μεταξύ των δύο αλγορίθμων, η οποία όμως μειώνεται καθώς αυξάνεται ο αριθμός των κόμβων του δικτύου.

Σημείωση

Αρχικά υποθέσαμε ότι ο βέλτιστος τρόπος ανάθεσης των προτεραιοτήτων είναι εκ των προτέρων γνωστός και στη συνέχεια προχωρήσαμε στην εύρεση των βέλτιστων σπασιμάτων για την δεδομένη ανάθεση προτεραιοτήτων. Αυτό είναι αρκετό για την περίπτωση του Objective1 καθώς ο μc rule υποδεικνύει τη βέλτιστη ανάθεση προτεραιοτήτων. Ωστόσο κάτι τέτοιο δεν ισχύει για την περίπτωση του προβλήματος αυτού. Για παράδειγμα για $\boldsymbol{\lambda}$=[0.95 0.78 0.97 0.7] ο ορισμός των προτεραιοτήτων ως node1 > node4 > node2 > node3 δίνει καθυστέρηση 4.3162 ενώ η ακολουθία που προτείνει ο μc rule έχει καθυστέρηση 4.5733. Επομένως, καθώς δεν υπάρχει κάποιος προκαθορισμένος βέλτιστος τρόπος ανάθεσης προτεραιοτήτων απαιτείται η δημιουργία ενός ευρεστικού αλγορίθμου που σε συνδυασμό με τους αλγορίθμους εύρεσης των βέλτιστων σπασιμάτων θα βρίσκει την βέλτιστη λύση του προβλήματος.

Κεφάλαιο 6 – Μελλοντικές επεκτάσεις

Στην παρούσα εργασία μελετήσαμε την βεβαρημένη καθυστέρηση απόκτησης ενός αρχείου, η οποία εξαρτάται από τον διαχωρισμό των αιτήσεων στους κόμβους άλλα και από την πολιτική εξυπηρέτησης κάθε εξυπηρετητή. Θεωρήσαμε ότι οι παράμετροι b_i είναι σταθερές κατά τη διάρκεια της προσομοίωσης, γνωστές σε όλους τους χρήστες, και ότι εκφράζουν τη σημασία κάθε χρήστη για το δίκτυο. Επομένως, τα βάρη αυτά θα μπορούσαν να είναι οι καθολικές τιμές reputation κάθε χρήστη στο σύστημα, που καθορίζονται με κάποιον εκ των γνωστών αλγορίθμων διαχείρισης εμπιστοσύνης.

Ωστόσο, σε πραγματικά συστήματα, η καθολική τιμή reputation δεν είναι γνωστή στους χρήστες, αλλά ο καθένας έχει στη διάθεση του μια εκτίμηση των τιμών αυτών, που μεταβάλλεται με την πάροδο του χρόνου. Θα ήταν λοιπόν ιδιαίτερα ενδιαφέρον να εξετάσουμε την απόδοση των προτεινόμενων αλγορίθμων στην περίπτωση αυτή και την δυνατότητα ενσωμάτωσης αλγορίθμων διαχείρισης εμπιστοσύνης στο παρόν σύστημα.

Επίσης, κρίνεται απαραίτητη μια διεξοδικότερη εξέταση του προβλήματος ελαχιστοποίησης του μέγιστου χρόνου ανάκτησης λόγω της σπουδαιότητας των πιθανών εφαρμογών του.

Τέλος, στο μέλλον θα ήταν δυνατή η μελέτη των παραπάνω προβλημάτων για άλλα μοντέλα ουρών ή και για άλλες εκφράσεις καθυστέρησης.

Βιβλιογραφία

[1] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel and D. D. Yao, "Optimal Peer Selection for P2P Downloading and Streaming".

[2] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer- to -Peer Networks".

[3] S. Sanghavi, B. Hajek and Laurent Massoulie, "Efficient Data Dissemination in Unstructured Networks".

[4] J. Sethuraman, M. Squillante, "Optimal Stochastic Scheduling in Multi-class Parallel Queues".

[5] S. Borst, "Stochastic Dynamic Programming & Control of Queues", http://www.cwi.nl/~sem/LNMB/2004/.

[6] J. Liang, R. Kumar, K. Ross, "The KaZaA Overlay: A Measurement Study".

[7] R. Kumar, K. Ross, "Peer-Assisted File Distribution: The Minimum Distribution Time"

[8] J. Mundinger, R. Weber and G. Weiss, "Optimal Scheduling of Peer to Peer File Dissemination"

[9] *J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips,* "The Bittorrent P2P File-Sharing System: Measurements and Analysis"

[10] D. Bertsimas, "The achievable region method in the optimal control of queueing systems; formulations, bounds and policies"

[11] B. Mortazavi and G. Kesidis, "A model of a reputation system for incentive engineering"

[12] R. Ma, S. Lee, John, C. Lui, D. Yau, "Incentive Resource Distribution in P2P Networks"

[13] K. Lai, M. Feldman, I. Stoica, J. Chuang, "Incentives for Cooperation in Peer-to-Peer Networks"

[14] L. Lai and H. El Gamal, "The Water-Filling Game in Fading Multiple Access Channels"

[15] Τυχογιώργος Γιώργος, "Διάδοση εμπιστοσύνης με μεθόδους βασισμένες στη φυσική", Διπλωματική εργασία